Intelligent Systems Technology, Inc.

March 25, 1995

Defense Technical Information Center
ATTN: Acquisitions/OCP
Cameron Station, Bldg. 5
Alexandria, VA 22304-6145

Intelligent Systems Technology, Inc. is pleased to submit this SBIR Phase I Final Technical Report for Contract No. DAAH01-94-C-R291 (Sequence Number A002).

Sincerely,

Azad M. Madni

Dr. Azad M. Madni
President and Chief Executive Officer

3100 Dannyhill Drive
Los Angeles, CA 90064
Tel. and Fax (310) 838-4883
E-mail: istinc@aol.com

"A Scalable, Customizable MCM Design Process Manager"

Scientific and Technical Report
Phase I Final Technical Report

March 23, 1995

Sponsored by: Advanced Research Projects Agency (DOD)
ESTO: Dr. Nicholas Naclerio

ARPA Order 5916
ISTI-FTR508-01

Issued by: U.S. Army Missile Command

Under: Contract No. DAAH01-94-C-R291

Principal Investigator: Azad M. Madni, Ph.D.
(310) 838-4883

Reporting period: August 23 to March 23, 1995

Effective Date of Contract: August 23, 1994

Contract Expiration Date: March 23, 1995

Accesion For		
NTIS CRA&I		X
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

Intelligent Systems Technology, Inc.
3100 Dannyhill Drive Los Angeles, CA 90064 Tel. and Fax: (310) 838-4883 E-mail: istinc@aol.com

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISTI-FTR508-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Intelligent Systems Technology, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Missile Command AMSMI-RD-PC-GY
6c. ADDRESS (City, State, and ZIP Code) 3100 Dannyhill Drive Los Angeles, CA 90064	7b. ADDRESS (City, State, and ZIP Code) Redstone Arsenal, AL 35898-5280
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Advanced Research Projects Agency ESTO: Dr. Naclerio 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 3701 North Fairfax Drive Arlington, VA 22203-1714	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Phase I Final Technical Report "A Scalable, Customizable MCM Design Process Manager"

12. PERSONAL AUTHOR(S)

Madni, Azad M. and Madni, Carla Conaway

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Phase I Final	FROM 94 Aug 23 TO 95 Mar 23	95 Mar 23	76

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Collaborative Design; Process Management; Integrated Product-Process Representation; Design Automation; Multi-perspective Visualization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Despite the fact that multi-chip modules (MCMs) are a critical dual use technology, the rapid commercialization of MCM-based systems depends on achieving an order of magnitude reduction in NRE costs and development time as well as assuring first pass success in MCM design. Design process management is one key requirement to achieve dramatic reduction in cycle time. Phase I of this effort was devoted to creating a system concept and detailed design for a MCM Design Process Manager (MCM-DPM), a software tool for managing collaborative design within a distributed heterogeneous design environment. This report presents the architecture, key components, functionalities, usage concept prototype, and implementation plan MCM-DPM. The unique aspects of the MCM-DPM include: an integrated product-process representation, process tailoring to an organizations "best practices" and existing EDA environment; multi-perspective design entry and process visualization, dynamically defined process flows, and design process tracking and guidance over multiple heterogeneous platforms.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Azad M. Madni	22b. TELEPHONE (Include Area Code) 310-838-4883 / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 — Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

We are indebted to Dr. Nicholas Naclerio for his guidance, encouragement, and support of this project. We have thoroughly enjoyed the many working sessions at Hughes Microelectronics Division with Mr. David Zarnow and Mr. Robert McBride. They have provided invaluable insights into the MCM design process management problems and worked with us from the very beginning to develop and refine a comprehensive set of requirements for the MCM Design Process Manager. Their continuing support of our work is much appreciated.

MCM DESIGN PROCESS MANAGER: EXECUTIVE SUMMARY

Overview

MCM Design Process Manager (MCM-DPM) is a scalable, customizable MCM design process management software that is currently being developed under a Phase I SBIR award from the Advanced Research Projects Agency. The target application is Multichip Module design, the target insertion environment is Hughes Newport Beach, and the initial group of end users include Hughes, Motorola and IBM, the major contractors on the ARPA-sponsored Application-Specific Electronic Modules (ASEM) Program. The overall goal of the MCM-DPM is to achieve dramatic improvement in MCM design cycle time while also achieving MCM design quality objectives.

Motivation

As EDA environments evolve from managing only tools and data to managing the design process itself, the concept of design flow has become a central issue. Design flow describes the sequence of operations required to achieve design goals. However, most flow-based approaches are limited in that they: (a) involve a fixed sequence of pre-specified operations; (b) restrict designers to using only those flows, and (c) "hardwire" specific EDA tools to the flows. These artificial and arbitrary constraints not only fail to reflect realworld design processes but also stifle designer creativity and flexibility. This recognition coupled with the market demand for a flexible design process management capability provided the impetus for the MCM-DPM.

Issues

There are several issues that have to be dealt with in creating a scalable, customizable MCM-DPM. Table E-1 presents a summary of the major issues.

Table E-1.
Key Issues

• Representing the design problem – syntactically transparent and semantically rich – allow for monitoring, tracking, and measuring progress on multiple products and their respective design processes • Capability for creating and managing dynamic flows – avoiding unrealistic "flow straight-jackets" • Balancing designer creativity with management control – multiple entry perspectives – multiple entry and exit points – multiple viable options at every step • Tracking design process on multiple heterogeneous platforms – remote tool invocation, operation, termination – combination of UNIX, NT, Windows platforms • Tailoring requirements – company best practice, organizational structure, legacy data bases, EDA toolkits, computing platforms	• Platform strategy – targeting availability on Windows, Windows NT, and UNIX • MCM design segment selection – target site capabilities and host environment • Process tailorability to different client processes and business practices – supportable at Hughes in terms of available toolkit and measurements • Persistent state – survive system "crash" – resume work where you left off upon "logon" • Schedule and cost controls – earned-value tracking – activity-based costing • Standards compliance – MOTIF, CORBA 2.0, CFI – Windows, OLE 2.0, CFI

Approach

Our overall approach emphasizes innovation in MCM design process management while leveraging commercial-off-the-shelf EDA toolkits, repositories, and standards. Our "customers," i.e., Hughes Aircraft Company has been involved with us since project inception and is contributing to specifying the functionality and features offered in the MCM-DPM. The target toolkits are the Hughes DecoDesigner or Tanner Toolkit. The fact that these toolkits are on a common platform and environment facilitates our Phase II integration task. Our overall goal is to create a viable commercial product. Since realworld design environments consist of physically dispersed teams collaboratively working within a heterogeneous computing environment over a LAN or WAN, we have taken a scalable approach that will allow us to track the design process within a distributed heterogeneous environment. To this end, we are working with our customers in creating the MCM-DPM. The resultant prototype can be expected to reduce design cycle time while facilitating the jobs of designers and managers alike. The unique aspects of our technical approach are presented in the Table E-2 below.

Table E-2.
Unique Aspects of the Technical Approach

- Design problem schema guides dynamic flow generation.
- Integrated representation of product and process at multiple levels of abstraction assures product-process compatibility after introducing changes, facilitates explicit status tracking of resources and activities as well as the evolving state of the product, and allows accurate estimation of earned value.
- Strikes optimum balance between design flexibility and management control through multi-perspective entry into the design process and multiple options after completion of each activity.
- Supports multiple design objectives (e.g. brand new design, verification of a specific aspect of a design, resuming work on an unfinished "design object," and adapting a previous design).
- Allows design process tracking over a LAN/WAN within a heterogeneous design environment (multiple platforms, certain tools run only on certain platforms).

Product

The MCM-DPM, the end product will be written in C++. The system, which will be architected within a client-server configuration will be supported by a COTS repository and communication backbone. The commercial product, called ProcessEdge™/MCM will be available on Windows, Windows NT, and UNIX platforms. The MCM-DPM will facilitate the MCM design process by:

- managing roles, tools and data created during design;
- providing designers with multiple entry perspectives, i.e., goal, tool, data/product, or flow, before converging on a specific activity;
- guiding designers through the design process with dynamically defined flows;
- automatically collecting performance data (metrics);
- orchestrating and coordinating activities of a collaborative design team;
- computing earned value, and tracking cost and schedule variances;
- providing electronic forms (i.e., templates) for WA, ACO, ECR, sign-offs;
- incorporating e-mail communications for non-realtime communications;
- allowing designers to work with different COTS MCM design toolkits.

Payoffs

The MCM-DPM will produce several payoffs. For the EDA community, it will offer the first dynamic process flow-driven design process management system that tracks the process over a LAN/WAN within a distributed heterogeneous design environment. For the MCM and ASEM community, it will provide an effective means for significantly compressing design cycle time. For ARPA, it will make a significant contribution to the Multichip Integration and ASEM Programs while advancing the state-of-the-art in scalable process support technologies. For Intelligent Systems Technology, Inc. (ISTI), it will result in a commercially viable product that can be taken to several other vertical markets (e.g., manufacturing, banking, health care) requiring workflow management.

1. INTRODUCTION

1.1 The MCM Design Process Management Problem

The commercialization of Multi-chip Module-based products can be dramatically accelerated by reducing the cost and design cycle time of multi-chip modules (MCMs). These are two of the five key objectives of ARPA's Multi-chip Integration program which is directed to achieving a ten-fold improvement in NRE costs and cycle time while assuring first pass success in MCM design. The key drivers of NRE cost and design cycle time reductions are equipment, materials, processes, and design process management. Data collected at Hughes Aircraft Company on some recent projects indicate that design cycle time, i.e., elapsed time, tends to be roughly an order of magnitude greater than actual worked time. Analysis of such data revealed that dead times and waiting times were principally responsible for this difference. With the introduction of manual design process management practices at Hughes Newport Beach, design cycle time was greatly improved. Today, design process management has been identified as a key requirement in MCM design with the potential of dramatically improving team productivity and design cycle time with commensurate reduction in costs.

MCM design is a complex process with multiple design iterations and complex tradeoffs. MCM designs are driven by specific objectives or needs -- higher speed, smaller size, lower power and/or reduced cost -- that is not expected to be met with conventional packaging methods. Design is typically done by a collaborative group with geographically dispersed members. Members of the group need to concurrently access design data from their various locations. The EDA tools that are used during design come from different vendors. Designers tend to work on more than one design at a time. Different companies have their own "best practice" process which they follow during design. Occasionally, available EDA tools affect design flow in that some design tools are based on "postulate a layout-analyze-iterate" design paradigm while the more recent tools are based on "capture all known requirements-synthesize-refine" design paradigm.

Given the complexity of the design process, design process management has been understandably an elusive goal in electronic design automation. There are several technical deficiencies and economic issues that have to be successfully tackled before creating a successful design process management solution (Table 1-1).

Table 1-1.
Historical Challenges to Design Process Management

• Absence of an underlying process management methodology.
• Incomplete, representation of the problem.
• Lack of customizability in the process management software.
• Expensive runtime license of commercial-off-the-shelf repositories.
• Absence of relevant data collection procedures.
• Absence of standards-compliant communication backbone.

The methodological deficiency stems from the fact that the design process with conventional EDA tools tends to be ad hoc and primarily defined by the available EDA tools, i.e., tool-driven. As such, the design process with existing EDA tools continues to be implicit, hard-coded, and tool-driven, rather than explicit, reconfigurable, and task-driven.

The representation problem is concerned with the semantic "completeness" of the design problem representation. The explicit representation of all key aspects of the design process is required for process tailoring, tracking, measurement, intervention, and feedback. Relieving designers from having to deal with low level details of coordination and synchronization allows them to concentrate on the more innovative aspects of design. This recognition is a central theme in design process management.

Direct monitoring of the operations performed by the various tools (residing on different platforms) on the different "design objects" is key to tracking progress. Without this capability, tool invocation and termination events have to be used to infer the beginning and completion of activities. This approach is clearly inadequate because one has to infer the commencement/completion of an activity on the basis of tool invocation and termination, rather than direct knowledge of actual object manipulation.

Tracking "earned value" is another important aspect of design management. To date, project management tools are unable to provide up-to-date, accurate information about MCM design progress in terms of product evolution and process progress. As a result, design decisions are often based on outdated and unrealistic information.

Successful implementation of an MCM Design Process Manager (MCM-DPM) has the potential of significantly reducing both elapsed time as well as optimizing resource utilization -- the two key drivers of cycle time and cost. Recent advances in integrated product-process representation, dynamic process flow modeling, object-oriented modeling approaches, distributed object management environments, and metrology have made it feasible to implement design process management within electronic design automation environments. This recognition provided the impetus for the work reported in this document.

1.2 Project Objectives

The overall goal of this effort is to develop, evaluate, and commercialize the MCM-DPM, a scalable, customizable design process management software for use in Defense as well as commercial MCM design applications. The specific objectives of Phase I are to:

1. Specify a MCM design problem representation methodology.
2. Specify functionality, architecture, and metrics for MCM design process management.
3. Develop and demonstrate "proof-of-concept" prototype.
4. Create a Phase II implementation and transition plan.

The first objective is concerned with creating a scalable, semantically complete, integrated and customizable representation of the MCM design problem. This representation provides the foundation for design process management, as well as metrics specification. Specifically, the integrated representation is key to monitoring/tracking, querying and measuring the progress of individual designs when multiple "product' are being designed. Customizability is key to tailoring a reference model to each customer's organization and design practices. Finally, compliance with evolving standards (e.g., CFI) is important for software portability and third party tool integration. The second objective is concerned with the design of the MCM-DPM. A prerequisite to achieving this objective is capturing and analyzing existing manual design process management practices. Such analysis is central to identifying deficiencies in existing practices, and targeting improvement opportunities. A central issue in satisfying the second objective is creating an architecture that leverages standards-compliant COTS distributed object management tools and repositories. The third objective is concerned with communicating the design process

management concept to sponsors, customers, end users and developers to elicit meaningful feedback before embarking on Phase II. The fourth objective is concerned with developing a Phase II implementation plan that identifies the key tasks, their inter-relationships, as well as key demonstration milestones.

1.3 Related Work

Related work is ongoing both in academe as well as industry. The Bibliography includes a list of relevant work in this area. Minerva, a prototype design process manager developed at the Carnegie-Mellon University, offers four levels of abstraction (i.e., problem, CAD task, resource, component) to represent the overall design process. Minerva's overall benefit is realized when it is incorporated into a CAD framework (e.g., Odyssey), that supports resource and CAD task management.

Hercules, also from Carnegie-Mellon, is a task management software that employs task schema as the basis for dynamically defined process flows and multiple entry perspectives. However, these software prototypes are research environments with no commercialization plans. Neither tool has been applied to the MCM design management problem.

Knapp's Design Planning Engine (DPE) is a research prototype that generates plans for invoking CAD tools to realize design functions. DPE employs planning at the design task level, rather than at the process level. DPE does not offer process customization facilities.

The Engineering Process Management System (EPMS) from Syscon Corporation is another related development. EPMS is focused on workflow automation within an electronic shared-data information environment. Its ultimate goal is enterprise integration. Since EPMS is not specifically designed for design process management, its representation capabilities are not well suited for design.

Mentor Graphics Corporation (MGC) has developed PCB-Process Builder, a collection of product data management tools that provide printed circuit board (PCB) designers and project managers with off-the-shelf design process management capabilities. Based on MGC's WorkXpert workflow management family of products, PCB-Process Builder consists of PCB-Process Toolkit, FlowXpert and XpertBuilder. The PCB Toolkit contains MGC-recommended customizable concurrent board design process flows. FlowXpert is a multi-user application that offers a graphical view of design flows, task and data tracking, automated design steps, and a history of actions taken to complete the design. It allows project managers to view design progress and detect occurrence of potential data bottlenecks. XpertBuilder is a graphical drag and drop tool that enables engineers to define or modify process flows, and describe the relationships and dependencies between the flow steps. (PCB-Process Builder with three FlowXperts is priced at $30K. The products are also available individually with PCB-Process Toolkit at $5K, FlowXpert at $5K, and XpertBuilder at $15K.) Once again, the fact that design flow is prescribed a priori is a deficiency with this tool in so far as MCM design is concerned, given the rich set of tradeoffs and iterations that are the defining characteristics of MCM design.

1.4 Phase I Accomplishments

In Phase I, we created a complete set of requirements for the MCM-DPM with Hughes personnel. We then created a comprehensive model of the MCM design problem. We elicited the requisite knowledge to populate the model (see Appendix B for results of a sample elicitation). We then storyboarded, reviewed, and refined the usage concept with end users at

Hughes Aircraft Company. We also evaluated multiple implementation approaches before adopting one that met our requirements. Table 1-2 summarizes the technical accomplishments in Phase I.

Table 1-2.
Phase I Accomplishments

• Elicited a complete set of requirements for MCM Design Process Manager including the requirements for tailoring the product-process representation. • Created a semantically complete MCM design problem schema to capture problem domain constraints as well as guide the design process. • Created a design process management approach based on dynamically defined flow, multi-perspective design process entry, and multiple entry and exit points. – realtime instantiation, customization, and management of design process flows • Populated the problem schema with domain-specific as well as Hughes-specific data. • Evaluated and identified several COTS software packages in support of design process management (GUI, repository, object request broker technology for distributed computing systems). • Created concept of operation prototype in the form of a series of Windows '95 screens. • Developed a Phase II implementation plan.

1.5 Report Roadmap

Section 2 describes the system concept and functionality of the MCM-DPM. Section 3 presents the architecture of the MCM-DPM. Section 4 presents the MCM-DPM system implementation. Section 5 presents the Phase I Prototype in the form of Windows '95 screens with appropriate explanations. Section 6 presents the Phase II implementation plan.

2. SYSTEM CONCEPT AND FUNCTIONALITY

2.1 System Concept

Ideally, the design process and its management should drive MCM tool specification and development. However, in reality there are several different MCM toolkits already on the market. Hence, the design process management software should be capable of working with these toolkits regardless of the underlying methodology embodied in these tools. In general, MCM toolkits are based on one of two different *design metaphors*. The first is the traditional approach based on "postulate (a layout)-analyze-iterate." The tools based on this design approach tend not to concurrently optimize electrical, thermal and packaging tradeoffs. As a result, the process flow tends to have several iteration loops as and when conflicts and constraint violations are discovered due to interactions between electrical, thermal, and packaging considerations. The second approach is based on "known requirements capture-synthesize-refine" paradigm. In this approach, multiple factors involved in design are identified and considered simultaneously during design optimization. Tools patterned on this paradigm (e.g., Interconnectix tools) circumvent those design iterations that result from sequential consideration of design factors. In other words, the latter approach is based on concurrent optimization of tradeoffs while the former is based on sequential optimization. The MCM Design Process Manager (MCM-DPM) is being designed to work with either approach, i.e., it is methodology-independent.

The next important consideration has to do with *designer perspective*. Designers "enter" the design process from different perspectives before converging on and performing a design activity. The designer can undertake the design process with the intent to create a design from scratch, create a partial redesign, verify an aspect of the design, or complete the design of an unfinished component. The MCM-DPM will offer a common user interface to achieve these objectives. It will provide a graph-based, graphical visualization interface that conveys: (a) the status of activities, roles, tools; (b) the state of the design process in terms of roles, data, tools, and design objects in use at any point in time; and (c) the state of the evolving product.

The MCM-DPM employs an integrated representation of the design problem that relates design goals, process, product, tools, roles, and data. This representation serves as the reference model to guide the design process, provides the basis for constructing dynamic process flows, and enables the accurate computation of "earned value." Figure 2-1 provides an overview of the key components involved in the MCM-DPM system concept.

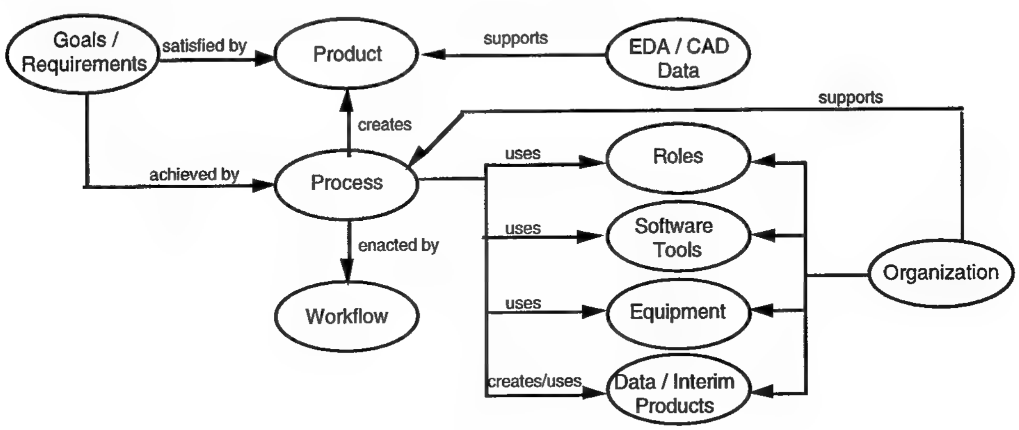


Figure 2-1. The MCM-DPM System Concept.

2.2 Realworld Design Scenarios

There are different realworld design scenarios that drive the requirements of the MCM-DPM. The designer may be undertaking a previously encountered problem or a brand new problem. For the former, the designer is interested in opportunity for reuse of past solutions. Occasionally legacy information in the form of previous designs or flows can be reused completely or in part, thereby simplifying the design process. For a brand new problem, the designer may enter and examine the design process from different perspectives before converging on an activity (Figure 2-2). To this end, MCM-DPM, will offer four different entry perspectives: goal/requirements, activity, data/product, and flow. When the designer enters the process from a goal, data/product, and flow perspective, he/she eventually converges on a specific activity and then selects an appropriate tool. If the designer picks a specific flow to follow, he/she could make minor modifications to the flow or its parameters. The resulting activity sequence is then pursued by the designer.

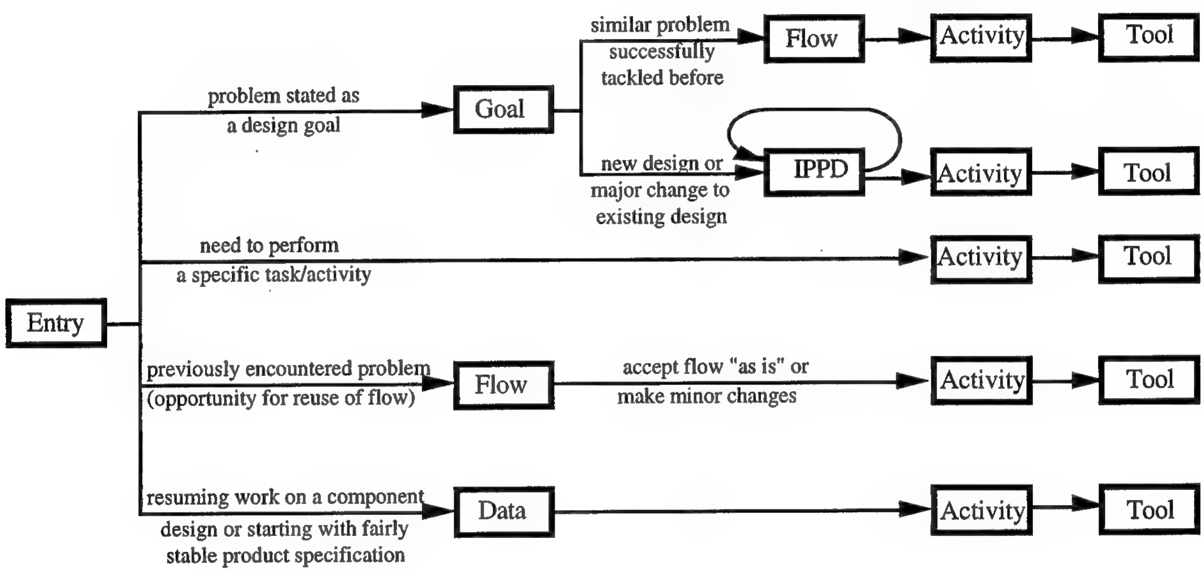


Figure 2-2. Entry Perspective is a Function of Initial Conditions

Table 2-1 compares and contrasts the different approaches in terms of their prerequisites, their applicability to the design process, and their appropriate context.

Table 2-1.
Design Perspectives

Candidate Approaches / Comparison Criteria	Goal-Driven	Tool-Enabled	Data-Driven	Plan/Flow-Enabled
Starting Point	designer selects a goal; goal is associated with a task in task schema	designer selects tool-entity or tool-instance	designer selects existing data, e.g., initial product spec, design object as starting point	designer selects a flow from a flow library
Design Context(s)	used when attacking a new design problem or subproblem	used when interested in performing a specific task or verifying a certain performance, e.g., thermal analysis	used when product spec is reasonably stable or when resuming work with a design object; reuse product	used when repeating a common design activity; reuse activity
Prerequisites	knowledge of goal; goal embedded in a task schema	availability of tool or tool instance	access to data associated with design object or product breakdown structure, e.g., pointer to data in CAD tool data base	required flow available in flow library
Comments	essentially solving the problem from scratch with or without legacy constraints	used to spotcheck or verify results of a specific process step	frequent starting point when work in progress with a specific design object	reuse of previously defined flows reduces time and incidence of errors

2.3 Initial Conditions

The initial conditions can dramatically change the design process. Initial conditions pertain to the state of the product as well as the state of its design and manufacturing processes. The possible initial conditions and the accompanying problem statement are shown in Table 2-2.

Table 2-2.
Initial Conditions Define the Problem Statement

Problem Type	Initial Condition	Problem Statement	Solution Type
I	Fairly stable product specifications	Create process for designing and/or manufacturing the product.	Process innovation
II	Initial product specs & process specs./tech. constraints	Create conflict-free final product and process design.	Concurrent product-process optimization
III	Design goal	Create a confict-free product and process design from scratch.	Invention

The majority of MCM design problem will fall under the category of a Type II problem, i.e., product design objectives exist in the form of initial product specifications, MCM tools exist with some default process sequence, and the design objective is to create an MCM design that satisfies product specifications using available tools in the proper sequence.

2.4 Product Functionality

Broadly speaking, MCM-DPM will be a distributed, integrated MCM design environment that will enable a group of designers to design MCMs rapidly and efficiently. Specifically, MCM-DPM will support:

- integrated product-process representation;
- dynamic instantiation and customization of the reference MCM product-process model;
- a collaborative design environment implemented within a platform-independent computer network for group coordination and cooperation;
- integration of MCM design tools from different EDA vendors;
- relevant standards such as CFI, CORBA 2.0, MOTIF/Windows.

2.5 Integrated Product-Process Representation

The basis for our MCM-DPM is an integrated product-process-based representation of the MCM design problem (Figure 2-3). This representation, in principle, consists of three interrelated parts: a product specification, a process specification, and a set of "integration" relations that establish the production and consumption relationships between components of the product and process specifications. The requirements for an integrated representation of the MCM design problem are presented in Table 2-3.

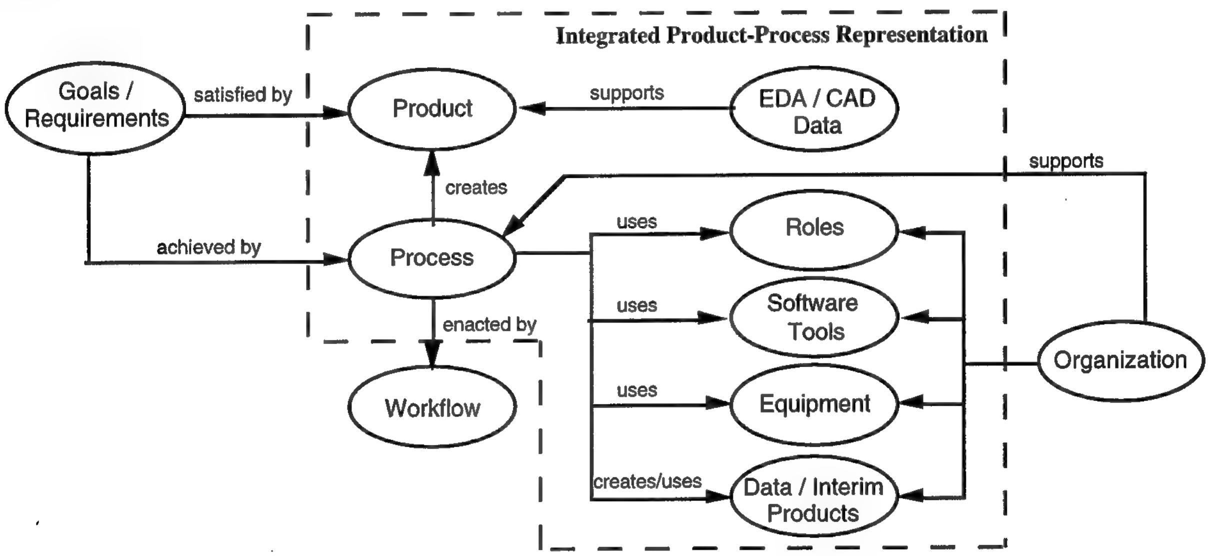


Figure 2-3. The Integrated Product-Process Representation Schema Provides the Basis for Creating Dynamically Defined Flows and Designer Guidance

Table 2-3.
Integrated Product-Process Representation Requirements

• Be easy to understand by MCM designers and managers.
• Model MCM process decomposition and product breakdown structure as well as their interactions.
• Fit within an integrated representation that related goals, product, process, roles, tools, data, and measurements.
• Provide the basis for constructing dynamic process flows.
• Describe resources (labor, technology), scheduling, costing, quality control, and others.
• Support visualization through multiple, interacting process enactment views (graphic or textual).
• Incorporate pointers that establish the semantic links to the real computer files that store the product or product component design, e.g., CAD file within an existing EDA/CAD tool.
• Provide the basis for MCM process management, costing, earned-value analysis, and enactment.
• Provide the basis for integrating multi-vendor EDA tools.

The single most important component of the integrated representation is the set of relations between products and processes. These relations, of an object-oriented class, serve a two-fold purpose: 1) They define the conceptual production and consumption relationships between the components of the design (i.e., product) specification and components of the process specification. For example, a typical production relation might contain a part, i.e., a product component, and a design activity, implying that the part is produced by that particular design activity. 2) They link and provide access to the product components and the process fragments. Through these relations, we can establish the semantic relationships between the product and process representations.

2.6 Model Tailoring

The models underlying the MCM-DPM are customizable to different organizations, design environments, computing platforms, management best practices, and legacy data and tools (Table 2-4).

Table 2-4.
Model Tailoring

• Different toolkits (Mentor, Cadence, Intergraph, Interconnectix, Tanner Research).
• Different platforms (UNIX, Windows, Windows NT).
• Different management practices (Hughes, Motorola, IBM).
• Different organizational structure (functional, product-oriented, process-centered).
• Different legacy designs and flows.

2.7 IPPR-Driven Design and Design Process Management

The impetus for our Integrated Product-Process Representation (IPPR) approach stems from the recognition that current MCM design approaches are oriented exclusively to individual designer support or manager support. For example, the user/tool-driven approach, which supports tool integration and automation, provides MCM designers with unlimited flexibility. However, the design process using this approach is invisible to the other members (i.e., other designers,

manager) of the collaborative design team. The lack of visibility into individual user activities with this approach makes it extremely difficult to manage and coordinate design activities. On the other hand, the process-driven approach emphasizes a formal and pre-defined MCM design reference model. This approach requires MCM designers to follow a predefined process during MCM design. While this approach facilitates management and control of the design process, MCM designers are forced to work with several predetermined constraints that no longer hold or cease to be applicable as the MCM design progresses. In addition, process inflexibility stifles designer creativity with this approach. Process management based on the integrated product-process representation approach overcomes the shortcomings of the preceding two approaches while capitalizing on their respective strengths.

The integrated product-process representation (Figure 2-4) consists of:

1) process model with scheduling, costing, and other pertinent information;
2) process flow model with decomposition and execution dependency ordering;
3) product model with decomposition;
4) product-process model interaction relationships;
5) multiple process representation interface;
6) graphical visualization perspectives.

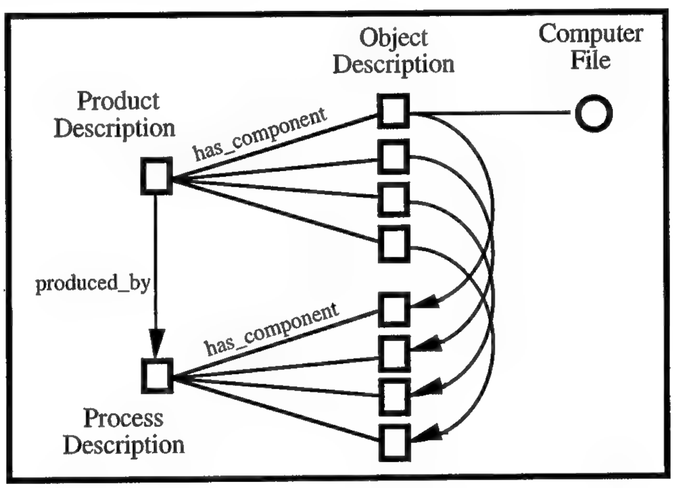


Figure 2-4. Integration Representation -- The Management Level

A key aspect of the integrated product-process representation is support for multiple dynamic instantiation and customization of the representation. Generally speaking, the design process model defined thus far consists of high level models that are largely independent of a specific product design. While these process models are useful for offline process guidance and conventional project management, they cannot be easily adapted for online realtime process management and execution. This is because the flow in these process models are generally defined without iteration loops and, therefore, are unable to reflect realworld design iterations. Also, these process models do not include any product description and, therefore, cannot support simultaneous performance and quality assessment/measurement of the evolving product design. As such, it is not uncommon for such predefined process models to become reference "shelfware," rather than an active guidance mechanism during design.

On the other hand, linking extraneous product information to a process model may make the process model unmanageable. Also, one can expect several design iterations and product-related

tradeoffs to occur before a product specification/design is finalized. These design iterations lead to an explosion in the number of process iterations and branches. Consequently, attempts to define a process model prior to achieving stable product specification/design are impractical.

To circumvent these problems, our solution is based on dynamic instantiation and customization of integrated product-process models for design management. It consists of the following steps:

1) *A set of model fragments consisting of product components and their design process flows is defined.* Since the product components selected at this point are primitive components that invariably are commercial-off-the-shelf items, their design process flows are relatively uncomplicated and, consequently, easy to manage. The outcome of this step is a component reference library that is (re)used during product design.
2) *To start product design, the product manager selects a high-level design process description* (based on, for example, the company's general design guidelines) and initiates the first process step which creates a top-level product design description. This description is communicated to the designers. The designers expand on the product design, and, at the same time, instantiate the design process model as additional product components are designed and/or selected. This dynamic instantiation continues as long as the product design continues to expand. New process steps are continually added or deleted depending on the specific changes in the product design.
3) *At any point during design, a complete process model can be constructed* and used to guide the design progress as well as estimate costs and compute earned value. It is important to realize that the process model is valid only as long as the prevailing product design is valid. When an existing product design is updated resulting in a new product design, the corresponding process model is also updated. When this happens, the management support tools (e.g., simulation, scheduling, cost estimation, earned value tracking) must be re-run to update the predictions/assessments.
4) Finally, *there are multiple entry perspectives into the design process* (Figure 2-2). For example, the designer can enter the design process from the product or the process point of view. A designer may choose the process point of view when he/she wants to follow a pre-defined process and continue to work on it. Alternatively, the designer may choose to start MCM design directly from the product point of view . In this case, the design process manager will show the current MCM design and allow the designer to work on his/her assigned part of the MCM design. In either situation, the design process manager will continually update the process and product representation based on the designer's selection and update of the MCM design. This approach reflects realworld situation which requires a critical balance between designer flexibility and management control. The next section provides a sample usage scenario.

This model-based approach reflects realworld design situations which demand a critical balance between designer flexibility and management control.

2.8 Dynamic Instantiation & Customization of the Reference Model

The concept of dynamic instantiation and customization of the reference MCM product-process model stems from the realization that realworld MCM design processes are intrinsically complex, highly dynamic and inexorably tied to the state and status of the product. Compounding the problem is the fact that they vary from company to company based on starting conditions and legacy requirements. As a result, attempts to define a priori a detailed, enactable MCM design process cannot succeed. What is needed is a dynamic approach that continually (and automatically) updates the design process description with changes in product design.

To this end, MCM-DPM will offer dynamic instantiation and customization with the following functionalities:

- create MCM reference models that are based on generic design process guidelines and generic MCM product design specification;
- instantiate the reference model to fit into a specific MCM design situation at the beginning of a project;
- expand the instantiated model as more detailed product design description is created. Where possible, the expansion will be automatic, i.e., new process fragments that are responsible for the expanded product design will be added to the instantiated process model without human intervention;
- when reference model exists for a new product design, the instantiated process model will be customized and the user will be asked to create a new reference model.

2.9 Collaborative Design Environment

A collaborative design environment for group coordination and cooperation, will be implemented within a platform-independent computer network. MCM-DPM will provide a collaborative design environment for MCM design with design process management capabilities. Compared to the existing EDA frameworks, MCM-DPM will offer several additional capabilities including:

- execution of the instantiated process model to guide the MCM design process;
- use of enactment progress information to coordinate individual designer activities, including notification of readiness of specific activities, propagating state/status changes, and updating product status;
- use of execution progress information for management including progress report generation, tracking, and schedule control;
- MCM-DPM will be implemented within a distributed computer network (LAN or WAN) to support geographically dispersed MCM design teams;
- MCM-DPM will be implemented without reliance on a particular computer platform, i.e., it will be platform-independent. To achieve this, COTS software components will be carefully selected for incorporation within MCM-DPM.

2.10 Integration of Multi-Vendor MCM Design Tools

EDA environments today have to employ third party tools to support the entire MCM design process. While various standards bodies are attacking the tool integration problem, progress has been slow. In light of this fact, MCM-DPM is being designed to offer a preliminary solution to the problem by providing a transparent integration mechanism that can achieve CFI objectives without user intervention. In other words, MCM-DPM will deliver CFI-compatible tool integration in a user-friendly fashion. Specifically, the product will support:

- flexible integration mechanisms for different types of EDA tools such as file-based or data-based tools,
- dynamic and logical invocation of EDA tools during MCM design (physical invocation means a designer has to know the physical path of an EDA tool as well as its invocation parameters),
- flexible reconfiguration of EDA tools when a designer chooses a different, but comparable EDA tool to replace the default tool predefined in MCM-DPM.

2.11 Compliance with Standards

Standards compliance (e.g., CAD Framework Initiative standards) is particularly important in MCM design since the design process involves several different EDA vendors. As far as standards are concerned, we have focused on two areas:

- platform-independent implementation so that MCM-DPM can be rapidly ported to a specific platform as and when the need arises;
- the software, hardware, API, and interfaces used in MCM-DPM will comply with existing and emerging industrial standards.

2.12 Performance Metrics

We have defined a set of metrics to evaluate the impact and utility of MCM-DPM in executing and managing the MCM design process. First and foremost, the MCM-DPM approach will be compared to current "best practice." The quantitative comparison metrics will be design cycle time, design iterations, resource utilization efficiency, cost, and accuracy of earned value tracking. The comparison will also be based on qualitative factors such as user acceptance, designer flexibility, and management control (Table 2-5).

Table 2-5.
Qualitative Comparison Metrics

• MCM designer acceptance.
• MCM product manager acceptance.
• Designer flexibility.
• Management control.
• Group coordination and collaboration support.
• Multi-vendor tool integration support.
• Distributed and multi-format MCM design data support.

The feasibility of MCM-DPM will also be established through integration with Hughes DecoDesigner Toolkit or the Tanner Toolkit. This integration will serve a three-fold purpose: (1) it will allow us to test whether MCM-DPM can successfully support tool invocation with the DecoDesigner Toolkit or the Tanner Toolkit; (2) it will allow us to test whether MCM-DPM can be used as an integration mechanism for loosely coupled MCM design tools. This could provide an alternative tool integration approach from that used in conventional EDA frameworks; and (3) the overall value of MCM-DPM will be determined by "loading" the Hughes MCM design process into the MCM-DPM, inserting the resultant software into an "instrumented" Hughes MCM design environment, and measuring improvement on key performance metrics.

3. MCM DESIGN PROBLEM-RELATED KNOWLEDGE STRUCTURE

The system concept and design of the MCM Design Process Manager (MCM-DPM) is based on several joint working sessions with subject matter experts at Hughes Microelectronics Division, Newport Beach, California. Our objective was to capture their design process, a representative breakdown of the MCM design, their tool suite, the various individuals (roles) involved in the design process, the various interim products created, design metrics, design management metrics, and earned value milestones. We were specifically interested in separating their "best practice" (e.g., reviews, when they were held, activity entry and exit criteria, and design completion criteria) form the generic MCM design process. Our intent was to be able to create the Hughes process from the generic MCM design process through process tailoring and object specialization.

Figure 3-1 shows the results of elicitation sessions devoted to capturing the MCM design problem (goals, process fragments, tools, roles, interim product, end product breakdown, earned value, design metrics, and process management). Figure 3-2 presents MCM Design Requirements Hierarchy. Customer requirements are often stated in these terms, i.e., the customer assigns values or constraints to this set of variables. Figures 3-3 and 3-4 present class-instance hierarchies for EDA tools and roles. Figure 3-5 presents a breakout of the end products, i.e., the footprint, nets, plots, files, and documentation.

The tools typically run on different platforms, e.g., the DecoDesigner-based on Mentor Graphics toolkit and supporting third-party tools run on UNIX platforms. Tanner Toolkit runs on the PC under DOS with Windows release expected shortly. The Hughes design team hopes to "mix and match" the different toolkits and tools as necessary. Their cycle time analysis has clearly indicated a pressing requirement for design process management.

	Process Management Metrics	Design Metrics	Design Goals	Process Fragments	Tools (PC) "Tanner Tools"	Tools (UNIX)	Role	Interim Products	MCM Breakdown Structure (end product)	Earned Value (in time) HDMI	Earned Value (in time) LTCC
MCM Design Problem Objects	M1 # of Components Captured	M1 # of Review Iterations by Assembly/Manuf.	G1 Satisfy Manuf., Assembly Rqmts.	PF1 Component Capture	Footprint Generator	MGC Librarian	Designer	IP1 Footprints (current design); Potential Future Standard Parts	Devices	40	20
	M2 # of Components Placed	M2 Total # of Placement Iterations	G2 Optimize Thermal and Critical Signal Rqmts. Concurrently	PF2 Placement	L-Edit L-Edit SPR	MGC Layout	Designer	IP2 Top Level Plots and Part Replacement	Device Location		
	M3 N/A	M3 # of Placement Iterations Due to Thermal Analysis	G3 Satisfy Thermal Rqmts.	PF3 Thermal Analysis	L-Edit Therm	MGC Auto-Therm	Mechanical Engineer	IP3 Thermal Report			
	M4 # of Errors	M4 # of Iterations Due to Pass Verification	G4 Verify Compliance with Manuf.	PF4 Top Level Verification	L-Edit DRC	Cadence Dracula	Designer	IP4 Go/No Go Flag with Potential Waivers			
	M5 % Routed	M5 Time to Perform Breakout Routing	G5 Preroute Nets to Bulk Routing Grid	PF5 Breakout Routing	not supported	MGC Layout and Hughes Glueware	Designer	IP5 Pre-routed Nets to Routing Grid	Part of Interconnect	N/A	30
	M6 % Routed	M6 Time to Route; # of Unroutes Left by Bulk Routing	G6 Satisfy Critical Signal Rqmts.	PF6 Critical Routing (manual routing)	not supported	MGC BS 500, Layout (or by hand)	Designer, Electrical Engineer	IP6 Routed Critical Nets	Part of Interconnect	15	15
	M7 % Routed	M7 Time to Route; # of Unroutes	G7 Route to 100%; Satisfy Critical Signal Requirements	PF7 Bulk Routing	V4 Router	MGC Layout	Designer	IP7 Routed Remaining Nets Except Power Ground	Part of Interconnect	25	15
	M8 % Planes Generated	M8 Power and Ground Bounce	G8 Achieve High Layer Coverage	PF8 Power Planes	"Area Fill" (limited support)	MGC Layout (typical), MGC Librarian, MGC Fablink	Designer	IP8 Generate Plane Layers	Power/Ground Planes	10	15
	M9 N/A	M9 # of Routing Iterations to Achieve Signal Integrity Rqmnts.	G9 Verify Critical Signal Requirements Are Met	PF9 Parasitic Analysis	T-Spice XTalk and Loss Check	Quad XTK or Quantic Labs, Green Field	Electrical Engineer, Designer	IP9 1) Signal Delay Report Can Be Back-annotated to Timing Verification; 2) Noise Report			
	M10 # of Errors	M10 # of Iterations to Pass Verification Criteria	G10 Verify Compliance with Manuf.	PF10 Design Verification	L-Edit DRC LVS	Cadence Dracula	Designer	IP10 Go/No Go Flag with Potential Waivers			
	M11 # of Necessary Drawings, Artwork, Manufacturing Data	M11 # of Foundry Failures	G11 Create Error-free Artwork	PF11 Fab Preparation	AutoCAD Interface (drawings) GDSII	MGC Fablink or CALMA	Designer	IP11 Assembly Drawings BOM, CALMA Stream File (GDSII), Probe Point File (Ed. text files)	Stackup and Artwork	10	5
Hughes Best Practice			G12 Satisfy Manuf., Assembly, and Thermal Requirements	PF12 Top Level Design Review			All Team Members	IP12 Action Items; Go/No Go Flags; Approved Placed Data Base			
			G13 Satisfy Manuf. and Electrical Requirements	PF13 Routing Design Review			All Team Members	IP13 Action Items; Go/No Go Flags; Approved Routed Data Base			
			G14 Satisfy All Requirements	PF14 Final Design Review			All Team Members	IP14 Action Items; Go/No Go Flags; Final Approved Data Base			
										100	100

Figure 3-1. Summary of Results of Hughes Elicitation Sessions

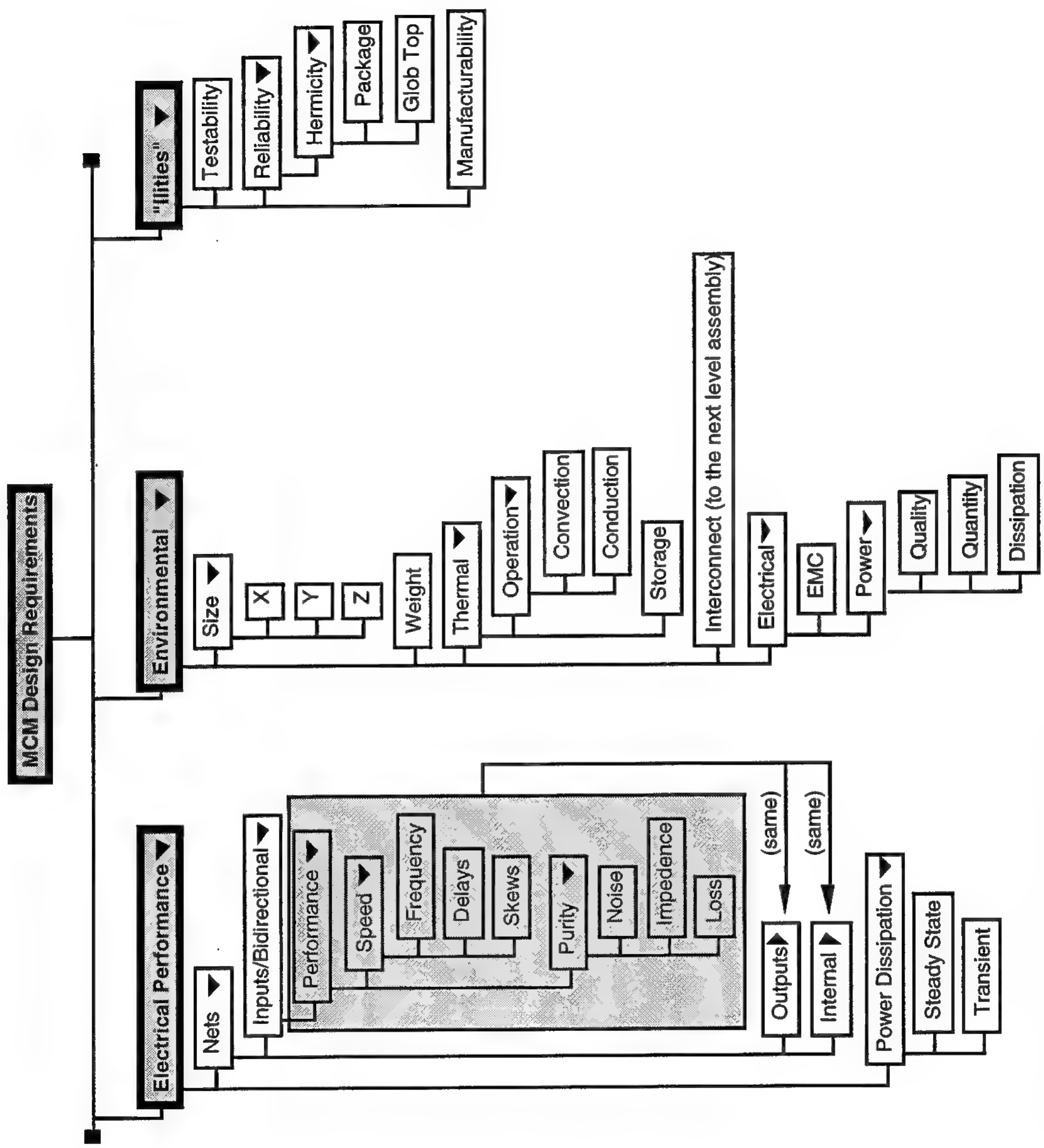


Figure 3-2. Design Requirements Hierarchy

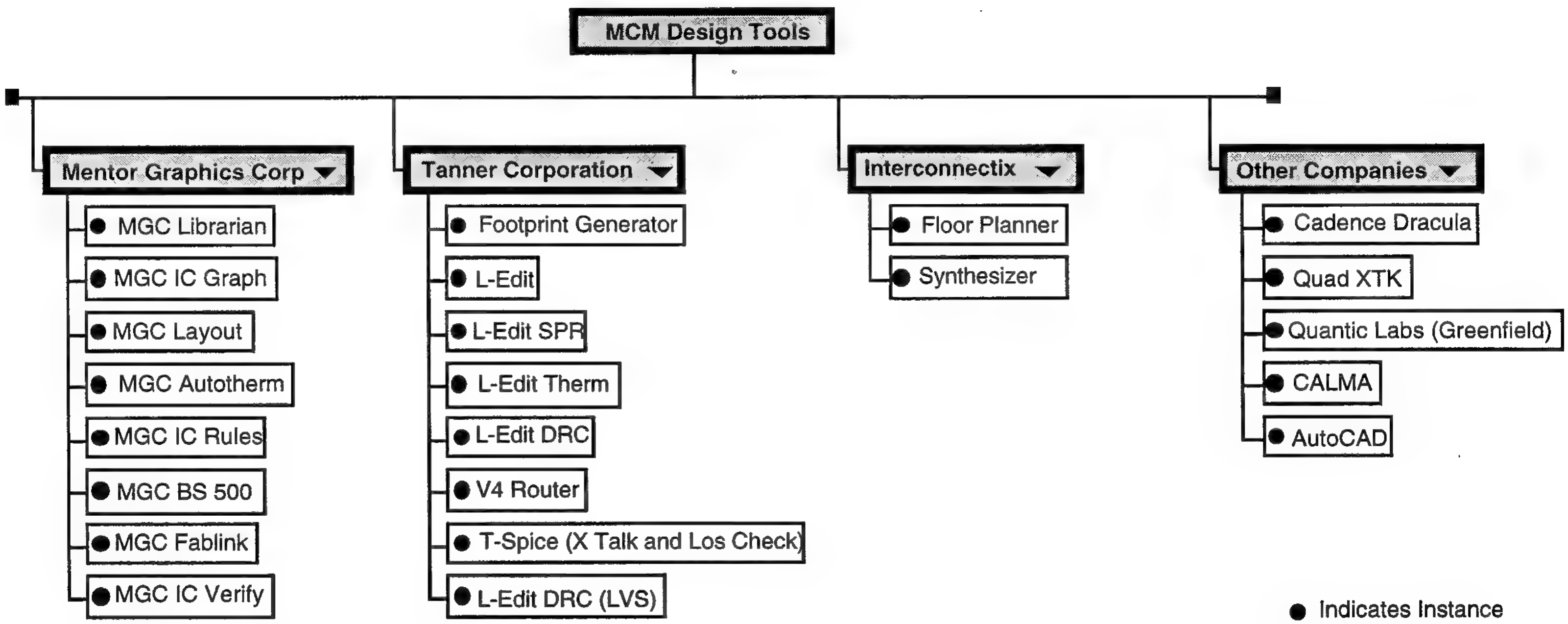


Figure 3-3. EDA Tool Hierarchy

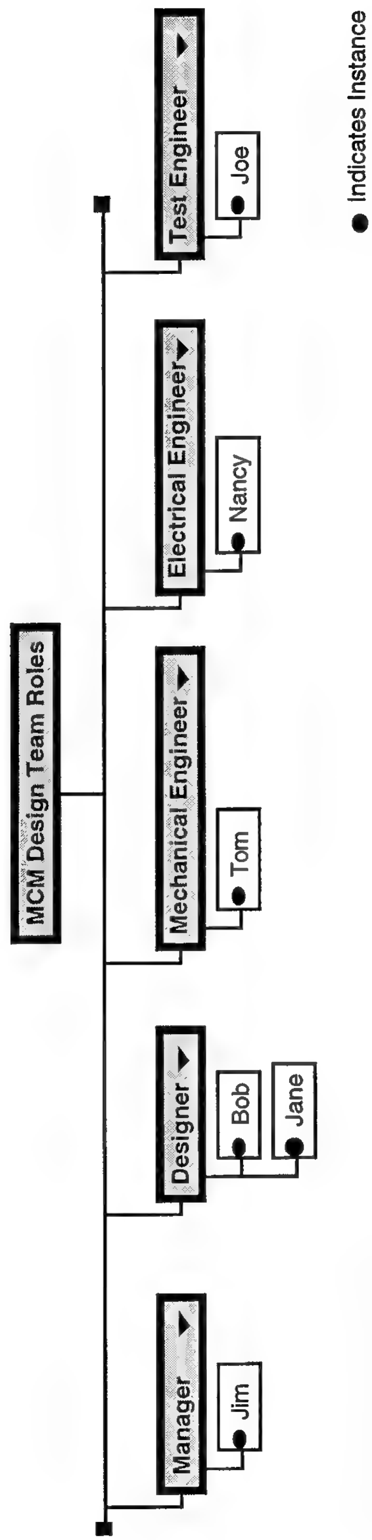


Figure 3-4. Role Hierarchy

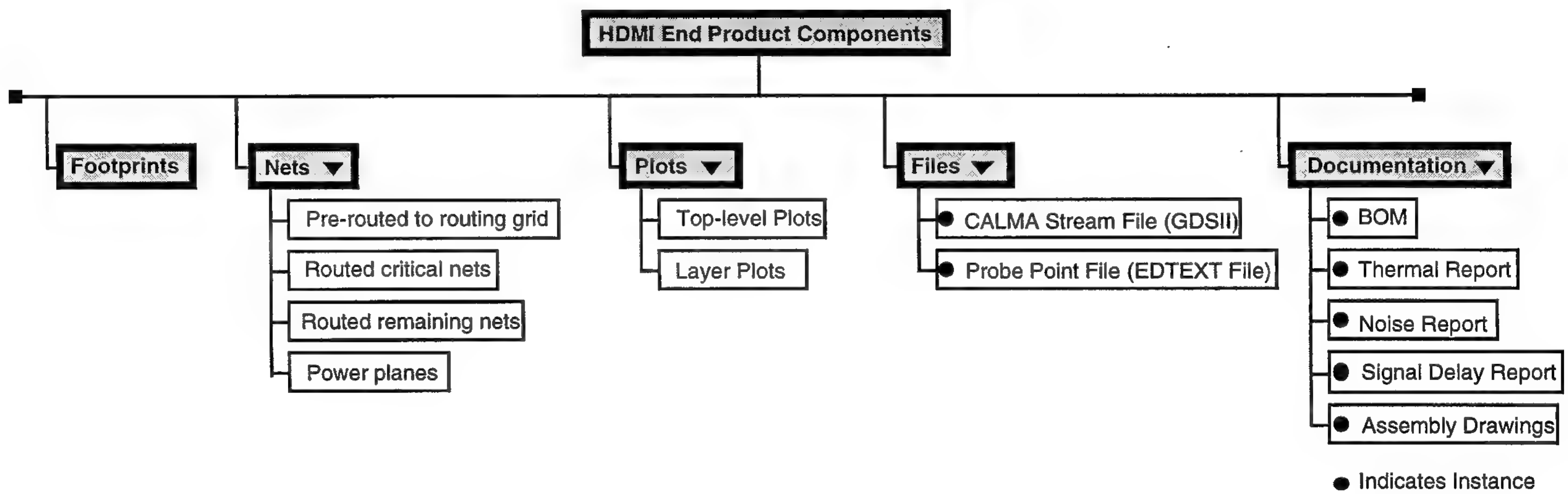


Figure 3-5. End Products Components Hierarchy

4. SYSTEM ARCHITECTURE

In the previous sections, we have presented the requirements and system concept for the MCM Design Process Manager (MCM-DPM). In this section, we present the system architecture and key components. The MCM-DPM is a process management tool that supports the concurrent execution of multiple MCM design activities using the modeling and execution approach described in earlier sections. In general, the MCM-DPM provides a distributed and integrated MCM design environment that enables a collaborative design team to design MCMs with faster cycle times and lower costs. Specifically, the MCM-DPM will support:

-- integrated process/product representation as described earlier;

-- dynamic instantiation and customization of a reference MCM design process model;

-- a collaborative environment for group coordination, which is implemented within a platform-independent computer network;

-- integration of EDA tools from different vendors to support MCM design;

-- in doing the above, the MCM-DPM will comply with the existing and forthcoming industrial standards (e.g., CORBA 2.0, CFI).

The MCM-DPM architecture is shown in Figure 4-1. Each component is described in the following paragraphs.

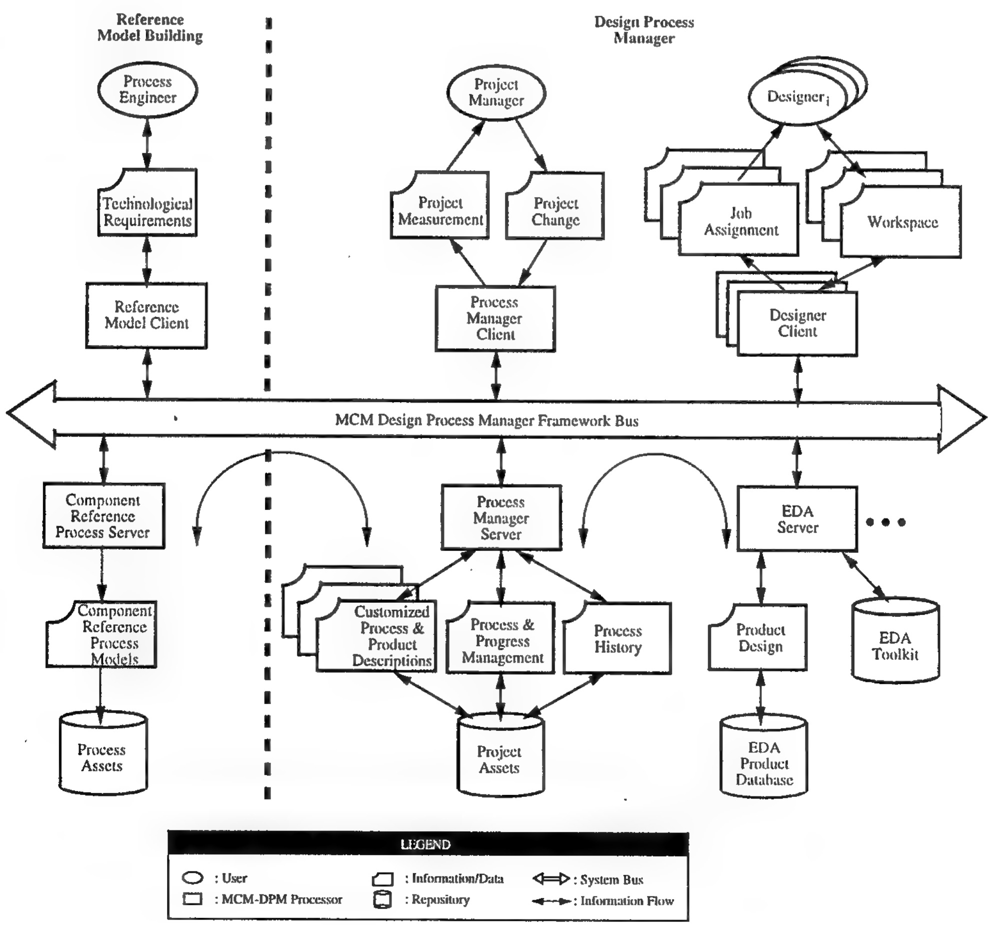


Figure 4-1. The MCM-DPM System Architecture.

4.1 Reference Component Model Server

The Reference Component Model Server provides modeling, customization, and tailoring of processes and products in support of design process management. First and foremost, it manages access to reference process/product component models that are stored in the Process Asset Data Base. Second, it supports concurrent manipulation of these product-process component models by a team of users. The kinds of manipulation include: definition and modeling, customization, and tailoring of product-process components. The Reference Component Model Server accesses the Process Asset Data Base for product-process component models. It serves the Reference Model Client and the Process Manager Client with reference product-process component models.

The Process Asset Data Base provides a repository for storing product-process component models. These component models describe various reference MCM design processes. These reference models vary in terms of the levels of specificity, the specific company policies, and/or the design technologies employed in the process. At the same time, the Process Asset Data Base provides semantic relationships to distinguish between and to organize different but similar reference MCM product-process models. Furthermore, the data base organization will support high-level reuse of these reference models. The access to the Process Asset Data Base is solely controlled by the Reference Component Model Server.

4.2 Reference Model Client

The Reference Model Client provides the user interface to define, customize, and tailor reference MCM design product-process component models, which can be instantiated to support a particular MCM design process instance. The Reference Model Client will support state-of-the-art approaches for model definition and customization through its graphical user interface (GUI). The model definition and customization capabilities will include different graphical views of a reference model (e.g., process decomposition tree, process flow, product decomposition), click-and-drop, and drag-and-drop. Users of the Reference Model Client are MCM design process engineers. These individuals have the responsibility to define/create MCM design process descriptions that are compatible with their respective company's policies, relevant MCM design technologies, and requisite levels of specificity. The inputs to the Reference Model Client are technological, organizational, and project-related. The output of the Reference Model Client is a set of MCM product-process component models that can be instantiated in realworld MCM projects. There could be multiple processes of the Reference Model Client at any point in time.

4.3 Process Manager Server

The Process Manager Server provides the major capabilities for MCM design management. These capabilities include dynamic product-process instantiation, process execution, project coordination, progress tracking, and measurement, and process history collection. It is the responsibility of the Process Manager Server to create a project, and execute it based on a customized and dynamic product-process model instance. The Process Manager Server controls the access to the Project Asset Data Base. It stores and retrieves customized process and product descriptions, process and progress management information, and collected process history to and from the Project Asset Data Base. It servers the Process Manager Client for the management information and the Designer Client for the design information respectively.

The Project Asset Data Base provides a repository for information about individual MCM design projects. This information includes customized product and process descriptions, process and

progress management information, and process history data. The use of the Project Asset Data Base is solely controlled by the Process Manager Server.

4.4 Process Manager Client

The Process Manager Client provides the user interface(s) for managing MCM design processes. MCM design management involves dynamic instantiation of process flows and product decomposition, tracking and control of process progress, and collection of process history data. The Process Manager Client invokes the Process Manager Server to accomplish these functions and to support its state-of-the-art GUI. Users of the Process Manager Client are MCM design managers whose responsibilities also include project management. Information regarding individual MCM design projects as well as their subsequent changes are the inputs to the Process Manager Client. The outputs of the Process Manager Client are project progress reports, measurement data, and historic data, i.e., audit trail. There could be multiple processes of the Process Manager Client at any point in time, with each process associated with a concurrently operating MCM manager.

4.5 Designer Client

The Designer Client provides user interfaces for enacting MCM design processes. The execution of a MCM design process involves: 1) following the guidance of the process model during MCM design, 2) continually expanding and refining the MCM product design, and 3) invoking MCM design tools on MCM design data (so-called tool integration). The Designer Client provides logical and collaborative workspaces (data and tools) for MCM designers to perform MCM design by following the instantiated process model. On the other hand, it is also a dynamic driver that automatically records the process progress, update the process state, and prompts other designers with proper job assignment. The Designer Client sends requests to the Process Manager Server to perform these functions. Users of the Designer Client are MCM designers. Inputs to the Design Client are design data and activities; its outputs are updates to both the MCM design (i.e., the product model) and the process model. There could be multiple processes of the Designer Client, each associated with a concurrently working MCM designer.

4.6 EDA Server

The EDA Server provides MCM design data and tool management. In fact, the EDA Server will be a COTS EDA toolkit integrated into the MCM-DPM. At this time, the candidate EDA toolkits being considered for integration are the Hughes DecoDesigner for UNIX platforms and Tanner toolkits for Windows platforms. These toolkits are currently in use in our target environment at Hughes, Newport Beach. The EDA Server is accessed by the Designer Client when a designer, executing a process step, invokes an EDA tool on a particular design object. At this time, the EDA Server provides a copy of the design object and "starts" the EDA tool which manipulates the object. The output of the EDA Server is a MCM design fragment that is stored in the EDA product data base.

The EDA Product Data Base provides a repository for storing MCM design data. The data is the output of a MCM design process. Depending on the EDA toolkit, there are different ways for implementing such a data repository. For example, for the Tanner toolkit, the data repository will take the form of a set of PC binary files with proprietary data format. In other instances, COTS or proprietary data bases could be more appropriate. To some extent, the MCM-DPM does not need to access the internal implementation details of the EDA Product Data Base Server. Rather, it needs the capability to access the various pointers to the different design objects used

and to transfer these pointers to the appropriate EDA design tools when needed. The use of the EDA Product Data Base Server will be solely controlled by the EDA Server.

4.7 EDA Toolkit Server

The EDA Toolkit Server provides a set of software tools to support MCM design activities. Most likely, it will be a collection of EDA tools from different EDA vendors. In this case, these tools will typically support different parts of the MCM design process. At the present time, we are focusing on the DecoDesigner or Tanner toolkit to demonstrate the feasibility of our MCM design process management concept. (Both these toolkits are available and in use in our target environment). To some extent, the MCM-DPM will not need to access the internal implementation details of the EDA Toolkit Server. It only needs to be able to invoke individual EDA tools and pass necessary parameters to it. The use of the EDA Toolkit Server will be solely controlled by the EDA Server.

5. SYSTEM IMPLEMENTATION

In this section we present our system implementation approach including standards compliance, repository strategy, platform strategy, integration strategy, and heterogeneity strategy. We then identify the different types of constraints and present our approach to handling them. We conclude with an evaluation of three different implementation options and present our rationale for the selected option.

The basic requirements for the MCM-DPM are:

a) integrated representation of MCM product and process;
b) execution and management of MCM design process, including process tracking, guidance, notification, and coordination within a distributed environment;
c) integration of EDA tools from different vendors;
d) support of heterogeneous MCM design environments.

5.1 Implementation Strategies

Our overall implementation approach is to leverage existing state-of-the-art software technology wherever possible to reduce development time and cost, and to comply with industrial standards in all areas that pertain to the MCM Design Process Manager (MCM-DPM). The individual strategies within this overall approach include standards compliance strategy, heterogeneity strategy, tool integration strategy, and repository strategy.

5.1.1 Standards Compliance Strategy

We will comply with industrial standards in all areas that concern the MCM-DPM including GUI, repository, tool integration, and communication network. The MCM-DPM provides process management services to MCM designers through a network of computers. The chosen network protocol will be EINet-based because it consists of standard network services in the EDA domain.

5.1.2 Integration Strategy

Our strategy is to support multiple cooperating users with EDA tools and data that can be encapsulated for integration and then presented to the users through synchronous and asynchronous user interfaces. To this end, we will leverage the capabilities of XShell, a COTS Distributed Object Management Environment (DOME) for tool encapsulation and integration.

5.1.3 Heterogeneity Strategy

One of the key requirements for the MCM-DPM is to support a heterogeneous design environment, in which different design tools run on different computing platforms. For example, the circuit layout tool under MS Windows on a PC creates a VLSI circuit that is fed to a thermal analysis tool under X-Window on a SUN workstation. It is crucial for the MCM-DPM to support such transparent transporting and sharing of data.

To this end, our heterogeneity strategy supports a multi-layered implementation: the first layer is the physical network layer where a set of computers of different platforms are networked to form a LAN (a WAN in the future). In reality, either NetWare, Ethernet, or TCP/IP network system can be used since they all support heterogeneous computer networks.

The second layer is the repository layer where data under the purview of the MCM-DPM are stored and accessed. Such a repository should support distributed and heterogeneous data

management. The candidate OODB we've selected for the MCM-DPM , i.e., ObjectStore, satisfies this requirement.

The third layer is the process server layer where process instances are enacted and monitored. At this time, we envision that there will be only one process server per EDA design environment. However, the server will be able to communicate with clients that reside on computers of different platforms within the environment. A very important capability of the process server is to choose appropriate computer platform for a given design activity and to shift the activity to the platform since the tool used in that activity runs only on that platform. The selected DOME satisfies this requirement.

The final layer is the client layer where actual design work is carried out. Since the clients reside on computer platforms from different vendors, it is very important to implement those clients for both UNIX workstations or PCs. To this end, our platform strategy is designed to ensure the simultaneous platform availability of the clients at very low cost.

Since the MCM-DPM implementation will be *platform-independent*, we have two choices for our platform strategy: (1) implement first on PC, then migrate to UNIX, (2) implement first on a UNIX workstation then migrate to PC. Each option is discussed next.

1) PC Implementation With a Pre-defined Migration Path to UNIX. This option calls for the use of Microsoft's Visual C++ in MS Windows or Windows/NT to implement the MCM-DPM. This implementation, as claimed by Microsoft, will support both Win16-type and Win32-type computers such as Motorola 80*86, Apple Macintosh, and DEC's Alpha workstation including operating systems such as MS Windows, Windows/NT, OS/2, Power PC, and Power Mac. Later, we will use a Visual C++-based migration tool, such as Bristol Technology's WIND/U, to port the software to UNIX workstations, (e.g., as SUN Sparc, SGI, HP 9000, and AIX 6000). Furthermore, since the two steps can be performed in parallel, the MCM-DPM can be made available on UNIX-based workstations shortly after the PC version. Compared to other implementation strategies, this strategy offers a high degree of platform availability at relatively low cost.

   We have elected to go with Visual C++ as our GUI builder as opposed to using platform-independent GUI builders such as XVT that help migration through multiple compilations of the same code on different platforms. We based our choice on the fact that since Microsoft is actively developing and marketing Microsoft Visual C++ as a multiple platform development toolkit, it is highly questionable whether these small GUI builder can survive. As such, taking Microsoft Visual C++ is a safe and low cost approach to platform-independent implementation strategy.

2) UNIX-based Downward Migration. This strategy calls for developing the software first on a UNIX platform such as SUN workstation (or LINIX on PCs) and then migrating it to Windows-based PCs. The main disadvantages of this approach are the high cost associated with development on UNIX workstations, if that is the starting point, and the incompatibility between UNIX workstations and PCs. On the other hand, UNIX workstations have the largest installed base in the EDA community. Also, initial development could be done on PCs under LUNIX and then migrated to Windows.

5.1.4 Repository Strategy

Repository is an integral part of the MCM-DPM for storing process, product, and project data. At the same time, our software has to interact with existing EDA tools that have their own data/product repository. As such, the repository requirements for the MCM-DPM are mixed. On the one hand, the MCM-DPM needs its own repository. On the other hand, it has to be integrated with repositories associated with EDA tools. Furthermore, these different repositories could reside in a distributed computer network.

Our strategy for the MCM-DPM repository will be executed in of two stages: During the first stage, we will select and use an object-oriented data base (OODB) for storing information that is created and used within the MCM-DPM. Examples of such information include: the integrated product-process representation, and some project-related information. After carefully evaluating existing OODB products on the market, we have chosen ObjectStore from Object Design, Inc. as our repository. For the past few years, ObjectStore has been the market leader in OODB arena and its benchmark has consistently outperformed those of its competitors. We will build object-oriented representation on top of ObjectStore and provide access mechanisms as part of MCM-DPM's data servers. The second stage is to selectively integrate legacy data repositories associated with EDA tools with ObjectStore. This stage involves building relationships and access mechanisms from our repository, i.e., ObjectStore, to the EDA repositories without compromising the contents of the EDA repositories.

5.2 Evaluation of Implementation Strategies

There are three possible implementation strategies for the MCM-DPM's process server (that includes primarily the process management engine):

1. In-house development.
2. Use of a COTS workflow management tool.
3. Use of XShell, a Distributed Object Management Environment (DOME).

In the following paragraphs, we review the pros and cons of these three approaches before adopting one as our implementation strategy.

1. In-house development. With in-house development, we can create a prototype that satisfies all the requirements. We have already discussed the key technical issues and presented the design of the MCM-DPM in earlier sections. So we could certainly pursue this route. However, the two main disadvantages of this strategy are 1) the high cost of the development and 2) the long delivery time. As a result, we have eliminated this option from further consideration.

2. Use of a COTS Workflow Management Tool. This strategy is based on selecting and integrating a COTS workflow management tool into the MCM-DPM. This workflow tool would serve as the process server engine. Workflow management products have existed on the market since 1994. Some of these tools have been successfully deployed in certain application areas. Examples of these products include Lotus Notes from Lotus Development, Inc., WorkMan from Reach Software, Inc., ProcessIt from AT&T, and InConcert from Xsoft.

   The two main advantages of this strategy are: 1) fast prototyping and demonstration by leveraging an existing COTS workflow tool; and 2) lower development costs. In this case, the main development issue is how to integrate the tool into the MCM-DPM to

support all necessary requirements. This strategy, therefore, does not require a major software engineering undertaking.

The disadvantages of this strategy include 1) dependency on the tool. Since the tool will provide process management capabilities, satisfaction of some MCM-DPM requirements will depend on the capabilities of the tool selected. For example, if the tool does not support distributed process tracking over a network, the MCM-DPM will be unable to provide this capability, a critical requirement for design process management. Most of the existing COTS workflow tools do not comply with the EDA standards simply because they are designed for application domains other than EDA. For example, there are COTS tools that support distributed workflow management. However, the communication protocols used in these tools are not CORBA compatible. As such, it will be very difficult for the MCM-DPM built on top of such tools to comply with EDA standards.

3. Use of XShell. XShell from Expersoft, Inc., is a DOME that supports the development of distributed software systems. The key features of XShell include: transparent distributed processing and communication, automatically generated remote object interfaces, dynamic CPU process creation and management, object-oriented model for parallel and asynchronous processing, simple and natural extensions to C^{++}, and CORBA compliance. Last October, ARPA awarded a two-year, \$24 million grant to a consortium, of which Expersoft is a member, to build a CORBA-compliant distributed object management infrastructure for commercial and defense applications.

   The implementation strategy, based on leveraging the capabilities of XShell, will create a high-level process management engine on top of XShell's distributed object management environment. This entails 1) building a distributed object hierarchy for the integrated MCM product and process representation, 2) enhance XShell's distributed CPU process creation and management capabilities to support high-level distributed MCM design process creation and management, and 3) build a non-programmer user interface to distributed process creation and management for MCM designers and managers. In sum, this strategy expands XShell's existing capabilities to high-level distributed process management.

   The advantages of this strategy include: 1) industrial-strength implementation. This strategy, if successfully executed, will create an industrial-strength, standards-compliant, general-purpose solution and implementation to the problem of distributed process management. Such a general solution is expected to greatly improve the productivity of MCM design. It is also applicable to other application domains (e.g., board design, manufacturing, complex systems development, ship building); 2) compliance with EDA standards. As indicated earlier, XShell is a front-runner among COTS products planning on delivering a CORBA 2.0 compliant solution. XShell also complies with other existing EDA standards such as object-oriented design, analysis, programming, and networking. By using a product with high-level standards compliance, the MCM-DPM will comply with all applicable EDA standards without engaging in a heavy implementation effort.

   The disadvantages of this strategy are: 1) significant software engineering effort. This strategy requires large-scale object-oriented programming effort. XShell is an object-

oriented programming tool, and C++ tool in particular. As with other C++ tools, it requires high-level programming and integration skills which typically take a long time to acquire. 2) preferred development on high-end computer hardware. At this time, XShell is most reliable on UNIX workstations. Therefore, if we adopt the XShell route, we believe it is a wise choice to start development on UNIX workstations, and possible a UNIX-based LAN. This option would require a somewhat larger computer hardware investment.

5.3 XShell-based Implementation

As indicated earlier, we currently plan on leveraging XShell for implementing a truly distributed process management mechanism. This section describes our technical approach for this strategy.

XShell is a distributed object management environment that provides distributed object definition and dynamic CPU process creation and management. Our strategy leverages these two capabilities. First, we will use the distributed object definition facilities of XShell to specify our integrated MCM product and process representation. However, since XShell is not an object-oriented repository, we will still use an object-oriented database to physically store the MCM product and process representation. This strategy requires the integration of XShell with an object-oriented database. Fortunately, XShell and ObjectStore has been integrated by Expersoft. We plan on leveraging this integrated software package. The combination of XShell and ObjectStore provides a sound basis for high-level distributed MCM design process management.

The key to this strategy is to expand XShell's capabilities to create and manage distributed objects (realized as CPU processes). At this time, XShell creates, tracks, and manages distributed objects within a distributed, heterogeneous environment, such as LAN of PCs, Macs, and UNIX workstations. Our strategy calls for a mechanism to create, track, and manage MCM design processes within a distributed, heterogeneous environment. Our approach, therefore, requires the expansion of XShell's distributed "CPU process" object creation and management facilities to manage high-level MCM design processes. Achieving this objective requires the cooperation from Expersoft's technical staff, which we have. In fact, we have started discussions with Expersoft regarding this required expansion.

For the MCM Design Process Manager to provide a total, recoverable view of the current (and previous) states in the development process, certain monitoring and distribution requirements must be met. First, whenever key distributed events occur, they must be monitored to allow for fault tolerance as well as to provide information for dynamically reconfiguring the distribution of distributed objects for optimal CPU/Network Bandwidth utilization. Second, all events that result in a change of state to the model must be logged so that the system can be "rolled back" to a previous state at any time. Third, system state changes in a distributed environment may cause a cascade of distributed actions to occur across the network. Most if not all functionality must be asynchronous in nature to prevent any process from being halted while waiting for the effects of all the distributed actions to be completed. The proper use of event logging and asynchronous communication (with event queueing) will allow applications to continue processing requests while a service is being performed for them elsewhere on the network.

Expersoft's XShell has built in tools/objects for event logging, asynchronous communications (with event queueing) as well as integration with OODB technology for the persistence of model/system state. A Monitoring and Distribution Object (MDO) may be built to include all of the functionality described above as regards to the monitoring and logging of model/system state.

But this MDO has no subcomponent/implementation specific behaviors. Once this Monitoring and Distributed Object has been written and tested, all other implementation specific objects dealing with every facet of the development process may be derived from the MDO and inherit its monitoring and distribution capabilities. Many of the MDO's capabilities may be inherited directly from Expersoft's XShell class library.

This design will allow experts in specific fields of the development process to build application specific objects without regard to monitoring and distribution functionality. They will simply inherit that functionality. Also, monitoring processes may be built based on MDO functionality that will automatically handle fault conditions based on system wide or application specific administrative policies. The monitoring processes may also be used to update, maintain, or view model/system state and/or functionality.

5.4 Constraint Capture and Handling

In this section we answer several key questions associated with capturing and handling the different constraints in MCM design process management. In the following paragraphs, we present the key questions followed by our response to each question.

1. *How are constraints captured including both allowable and disallowable process sequences?*

In the MCM-DPM system, constraints are conditions that are attached to the various relations between object classes. They determine whether or not a search is allowed to traverse the links where the constraints reside.

Currently the MCM-DPM supports four types of constraints:

1.1 Data Constraint for a task/activity. It specifies the input data needed to start a task/activity. Data constraints include data classes, data location, data format, and the tool class that manipulates the data. Semantically, a task/activity cannot be started until its data constraints are satisfied, i.e., the data are available for the task/activity.

1.2 Precedence Constraint for a task/activity. This constraint prescribes a partial order of tasks/activities (process steps) for process execution, i.e., one task can't start until another task finishes. However, precedence constraints alone do not determine the actual execution "flow" of the process. The execution "flow" depends on the satisfaction of <u>all</u> constraints including entry and exit criteria. Semantically, a task/activity can't be started until all its predecessors are complete.

1.3 Resource Constraint for a task/activity. This constraint specifies the resources needed to start a specific task/activity. Examples of resources include: people, data, tools, equipment, funds. Semantically, a task/activity can't be started until its resource constraints are satisfied, i.e., the informational and financial resources are available to undertake the task/activity.

1.4 Integration Constraint for a task/activity. This constraint specifies the necessary platform (hardware and software) requirements for starting a task/activity. It is actually a subclass of resource constraints. However, due to the fact that integration constraints are generally design-independent but platform-specific, it is separated out to support system integration activities. An integration constraint specifies the class of hardware and software platform (UNIX workstation with X-Windows) needed for

a task/activity to commence. Semantically, a task/activity cannot be started until it resides in a hardware and software environment that satisfies the integration constraint. Overall, the actual process flow is determined by the satisfaction of all existing constraints, not just process sequence (i.e., prescribed process precedence or ordering). It is entirely possible that occasionally some of the process sequence will be disallowed due to the fact that one (or more) other constraint is violated. In other words, process precedence or ordering is necessary but not sufficient for a process sequence to be considered admissible for execution.

2. *What is the strategy and protocol for achieving communication between Windows and UNIX platforms?*

There are two ways of approaching this problem. The first is based on loosely coupled communication. The second is based on tightly coupled communication. Each solution and its characteristics are discussed next.

1). Loosely Coupled Communication. We can build a domain-specific communication protocol on top of OODB for the MCM-DPM server and clients to communicate. However, this communication protocol will only support loosely-coupled communication, rather than tightly-coupled real-time data exchange. Generally speaking, tightly-coupled real-time data exchange requires sophisticated communication protocol such as OLE and CORBA. However, if the product is targeted to MCM design on a common platform, then we can get by with a loosely-coupled communication protocol. Such a protocol would entail:

   1) heterogeneous data communication between different computer platforms,
   2) tool invocation among computer platforms of same type, and
   3) automatic conversion of limited data formats.

   At this time, to create an advanced MCM-DPM prototype based on Tanner Toolkit or Hughes DecoDesigner, we don't see the need for dynamic data exchange and arbitrary data format conversion.

2). Tightly Coupled Communication. Tightly coupled realtime data exchange requires a more sophisticated communication protocol (e.g. OLE 2.0, CORBA 2.0-compliant). These capabilities are typically provided by DOME systems. An industrial-strength DOME that is expected to deliver CORBA 2.0-compliant solution in May 1995 is XShell from Expersoft. XShell supports distributed heterogeneous environments with capabilities for remote tool invocation and distributed CPU process creation and tracking. The latter capability provides the necessary communication "backbone" to construct and track distributed design processes. In addition, XShell (and distributed object management environments (DOMEs) of this genre) provide the necessary execution engine for design process management. The solution based on this approach not only fully addresses the MCM design problem involving heterogeneous platforms and tools within a LAN (or WAN), but is also scalable to other applications (e.g., board level design) and requirements of other related programs (e.g., RASSP).

In the interest of creating a scalable design process management environment that can support heterogeneous design environments and higher level applications, the tightly coupled communication approach is preferred.

3. *What is the strategy for automating data collection and keeping data up-to-date in the tool/database?*

There are two aspects to this problem: data update and Integrated Product-Process (data) Representation update. Data update will be under the purview of and controlled by EDA tools as it is today. In other words, the MCM-DPM will not interfere with existing EDA tools in so far as data management is concerned. For example, in the case of the Tanner Toolkit from Tanner Research, the tools manage the creation and access of data files that reside on a PC.

On the other hand, the update of data representation is part of the MCM-DPM's responsibility. There are three situations where data representation can be updated during process management:

3.1. Update data representation as the result of task completion. Once a task is complete, its result will be either a) a new data file, b) an updated data file, or c) update to the product database. In this case, the Integrated Product-Process Representation will be automatically updated as follows:

a) a new data object will be created with a pointer to the new data file; then appropriate links will be created to associate the data object with other objects in the representation; Also, the development state of the data object will be set to the initial value;
b) the development state of the data object that points to the updated data file will be updated;
c) the development state of the data object that points to the updated database section will be updated.

This set of steps handle the change propagation of the representation on the data side. There is also change propagation of the representation on the process side. The issue here is to update the development state of the process that is being enacted. When a task has just been completed, its state will be updated to 'complete'. Subsequently, the development states of the task's successors might be updated to 'ready' if all their constraints are satisfied at that point in time. The timing of change propagation is either when a task has just been completed or a task's constraints have been checked for the purpose of starting the task.

In addition, the collection of task completion information will also be used to create a "design history", which includes evolution of both design data and process execution. This historical information will support in the understanding of design rationale as well as continuous process improvement.

3.2 Update of data representation directly by designers. In this case, a designer modifies/expands the product representation in order to add a component to the MCM design (e.g., add a new circuit). In this case, both the data and data representation will be updated in a manner similar to that described above. The modification will be

directly on the product while the update to the process will be accomplished automatically by the MCM-DPM under normal circumstances.

3.3. Update of data representation directly by process engineers. In this case, a process engineer modifies/expands the process representation in order to, for example, add a new management subprocess. In this case, only the data representation will be updated since the data side (MCM design data) doesn't have to be updated. The modification will be directly on the process while the update to the data will be accomplished automatically by the MCM-DPM under normal circumstances.

To the extent possible, our design of the MCM-DPM will attempt to collect dynamic data and to maintain the integrity of the integrated representation. Clearly, our objective here is to free MCM designers from the burden of detailed data collection and to help them concentrate on the MCM design task.

6. USAGE CONCEPT DEMONSTRATOR

6.1 Overview

This section presents the usage concept in terms of a series of Windows '95 screens. The usage concept demonstrator that accompanies this report consists of a Macintosh-based PowerPoint® 3.0 presentation on 3.5" disk. The demonstrator is a color version of the screens presented in this section with explanatory charts introducting each segment.

The purpose of this demonstrator is to convey the MCM Design Process Manager (MCM-DPM) usage concept, key functionalities, design entry perspectives, dynamic flow construction, and designer and manager perspectives to the sponsor and end users for their evaluation and feedback. The results will be used to refine the usage concept.

The various screens are constructed to reflect the unique aspects of the MCM-DPM, specifically the representation of the design problem within a design problem schema and the use of the schema to create dynamic process flows. The design process in the MCM-DPM is incrementally defined as the designer engages in the design process. The different entry perspectives (i.e., goal, activity, tool, data/product, flow) are shown in the various user-system interaction screens (Figures 6-1 to 6-27). These figures collectively define the usage concept of the MCM-DPM. The following subsections present user-system interactions for the process engineer, the designer, and the project manager.

6.2 Process Engineer-System Interaction

This subsection presents the model building capabilities of the MCM-DPM. Specifically, it covers how the process engineer would enter organization-specific information required to undertake an MCM design project. It is the responsibility of the process engineer to create all elements of the Integrated Product-Process Representation (IPPR) to satisfy the customer's product requirements. The process engineer accomplishes this in one of two ways: 1) he creates the IPPR from scratch or 2) he tailors an existing IPPR that is stored in the MCM-DPM library. In either case, the MCM-DPM system provides full editing capabilities for the process engineer to specify and interrelate: project goals, design requirements, initial description of end product components, process activities and their functional relationships, data items, interim products, software tools, equipment, roles, and design modifiers. It also provides static and statistical model analysis capabilities so that the process engineer can ascertain the completeness and correctness of the entire IPPR. Figures 6-1 to 6-12 depict the process engineer-system interactions.

MCM-DPM Main Window. The main window of the MCM-DPM system (Figure 6-1) allows the user to immediately navigate to any other part of the system. The row of quick-buttons across the top of the window provide access to each of the elements of the IPPR with just one click. The bottom of the window gives the user instant access to local e-mail facilities that exist between the team members and the design team manager.

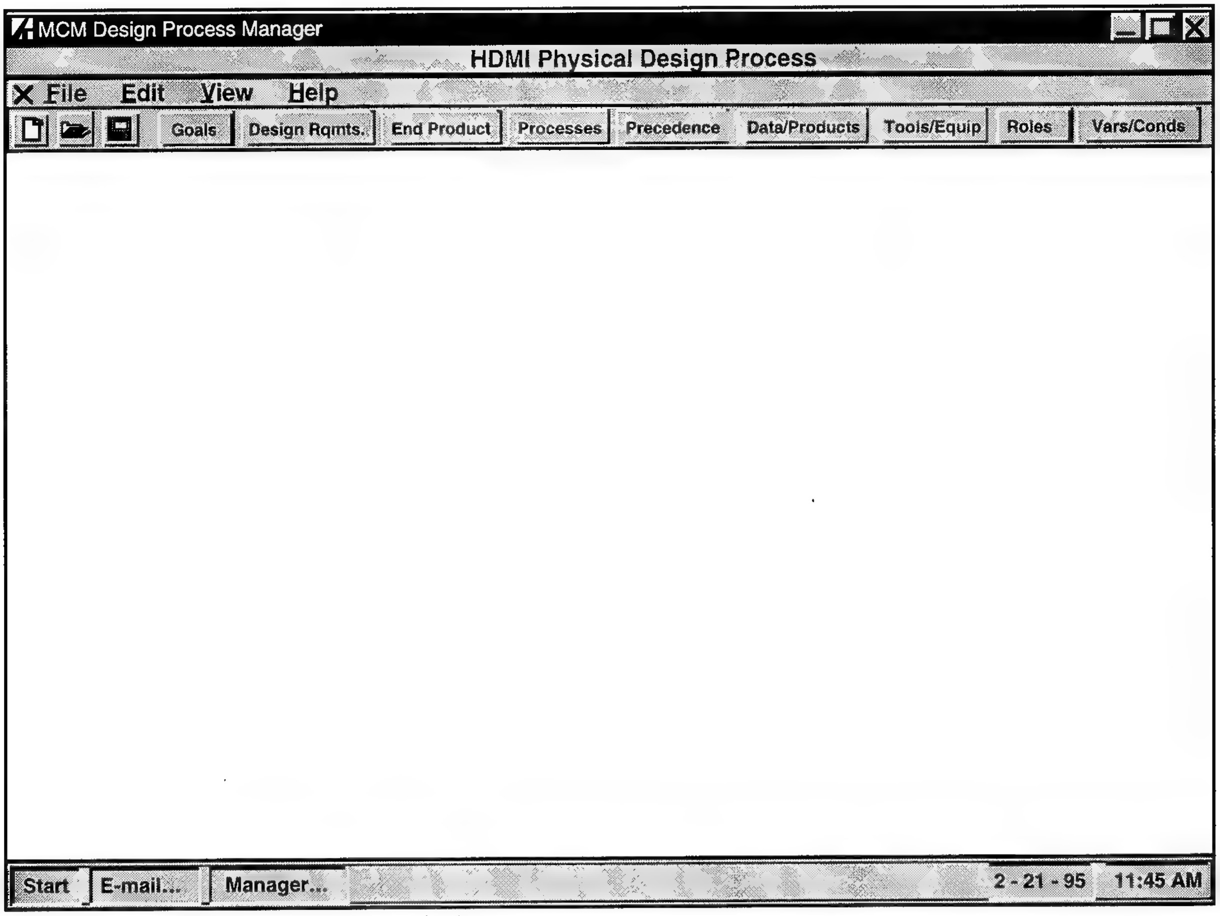


Figure 6-1. MCM-DPM Main Window

Building/Editing a Hierarchy. Many of the elements of the IPPR are represented as hierarchies. Specifically, the goals, design requirements, end product components, process activities, data items, interim products, software tools, equipment, roles, and design modifiers are all conveniently represented as hierarchies. Figure 6-2 shows how a completed hierarchy of process activities appears in the MCM-DPM. The higher level items can be "unfolded" to reveal the multi-level elements of the hierarchy. The window is scrollable both vertically and horizontally for viewing the entire hierarchy. The user can print a hard copy of the hierarchy, if needed.

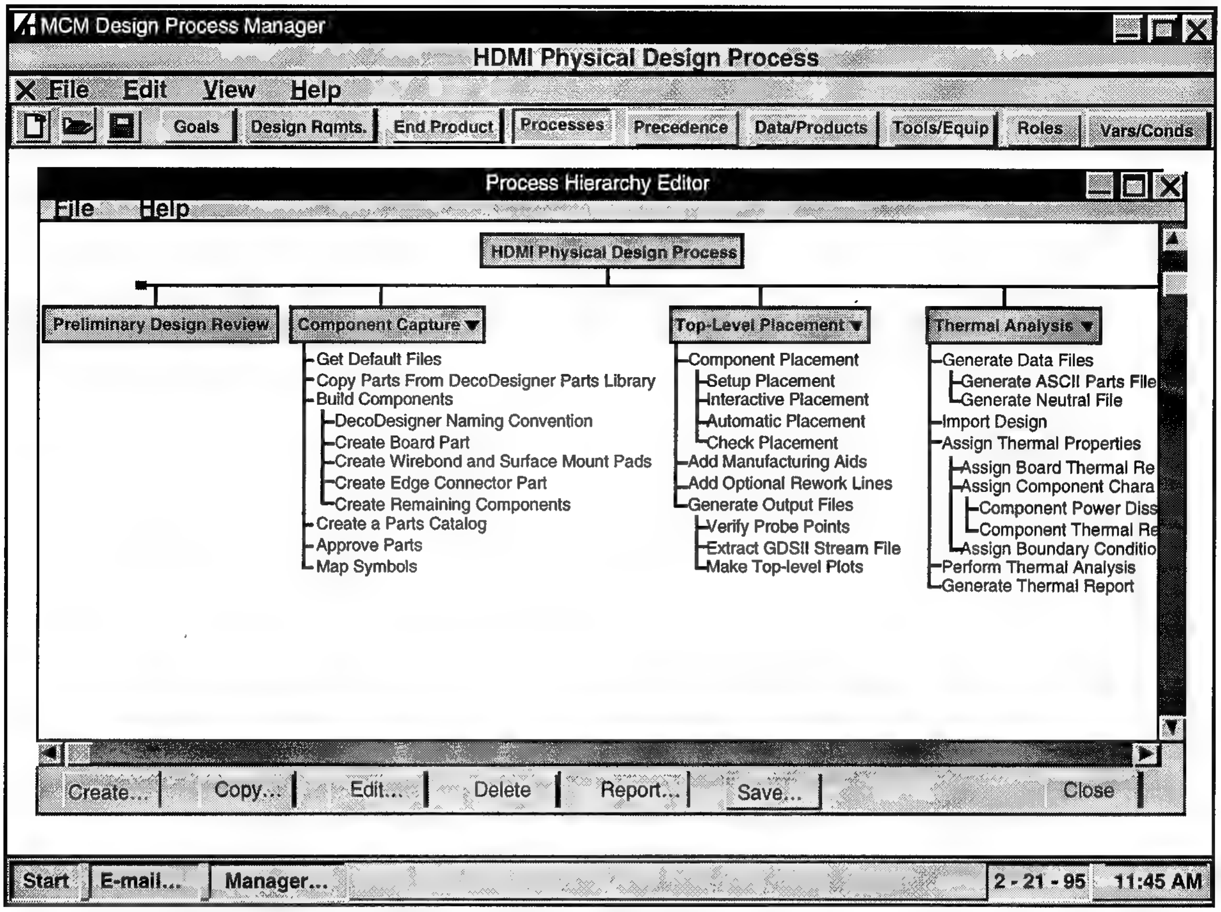


Figure 6-2. Process Hierarchy Editor

The hierarchy shown in Figure 6-3 (and all hierarchies throughout the MCM-DPM system) is created using an interactive series of windows to elicit the elements of the hierarchy and to specify their parent-child relationships. The user can create an activity by clicking on the "Create" button on the Hierarchy Editor Window. The following window will be presented. The activity name and description is specified and parent-child relationships to other existing activities can be created via this window.

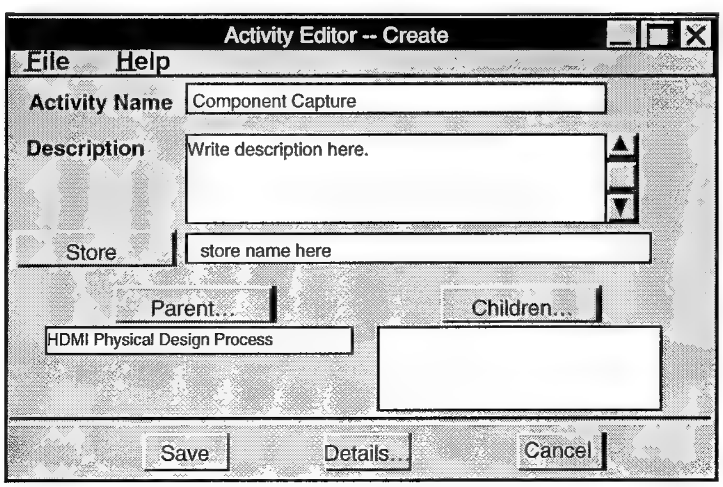


Figure 6-3. Create Activity Editor Window

When the user clicks on the "Children…" button, the "Children <activity name>" window is opened (Figure 6-4). This window allows the user to easily specify the children of an activity. It also allows the user to quickly create another activity to be also be included as a child of the originally-specified activity (by clicking on the "Create" button).

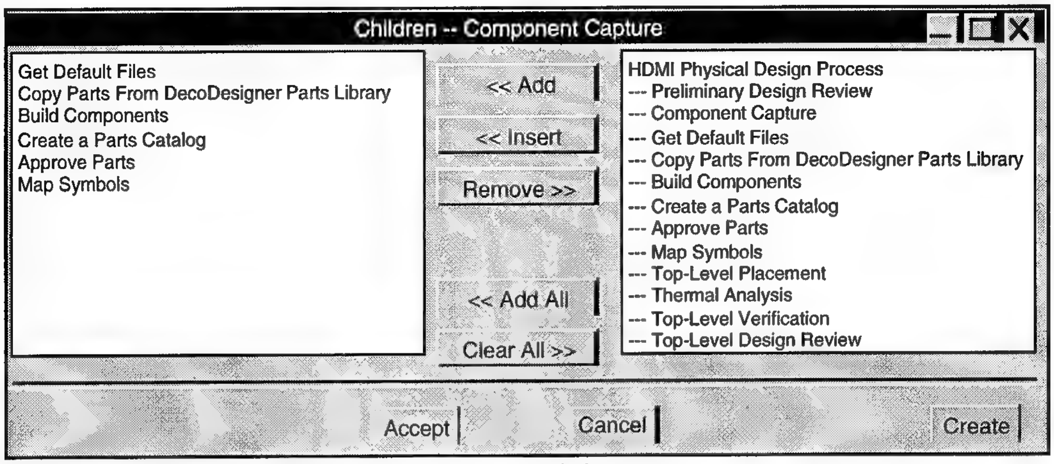


Figure 6-4. Window to Specify Children of an Activity

The results of the specifying the parent-child relationships as depicted in the previous figures is shown in Figure 6-5. In addition, these changes are instantly reflected graphically in the Hierarchy Editor window.

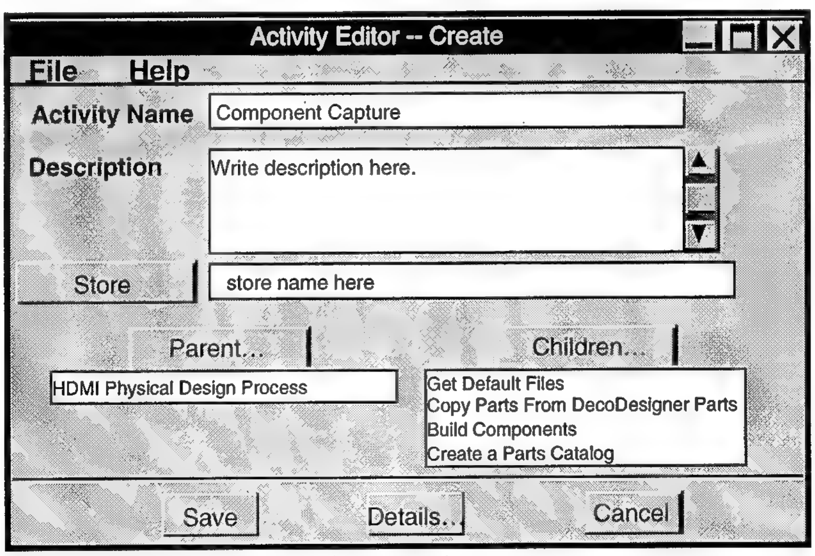


Figure 6-5. "Filled-In" Create Activity Editor Window

Building/Editing Activity Precedence Graph. Once the process hierarchy (of activities) is created, the process engineer can specify the precedence relationships between the various activities. The Activity Precedence Editor window is used for this purpose. Initially, the screen in this window shows the activities as specified in the hierarchy in a vertical column (no precedence relationships specified) (Figure 6-6).

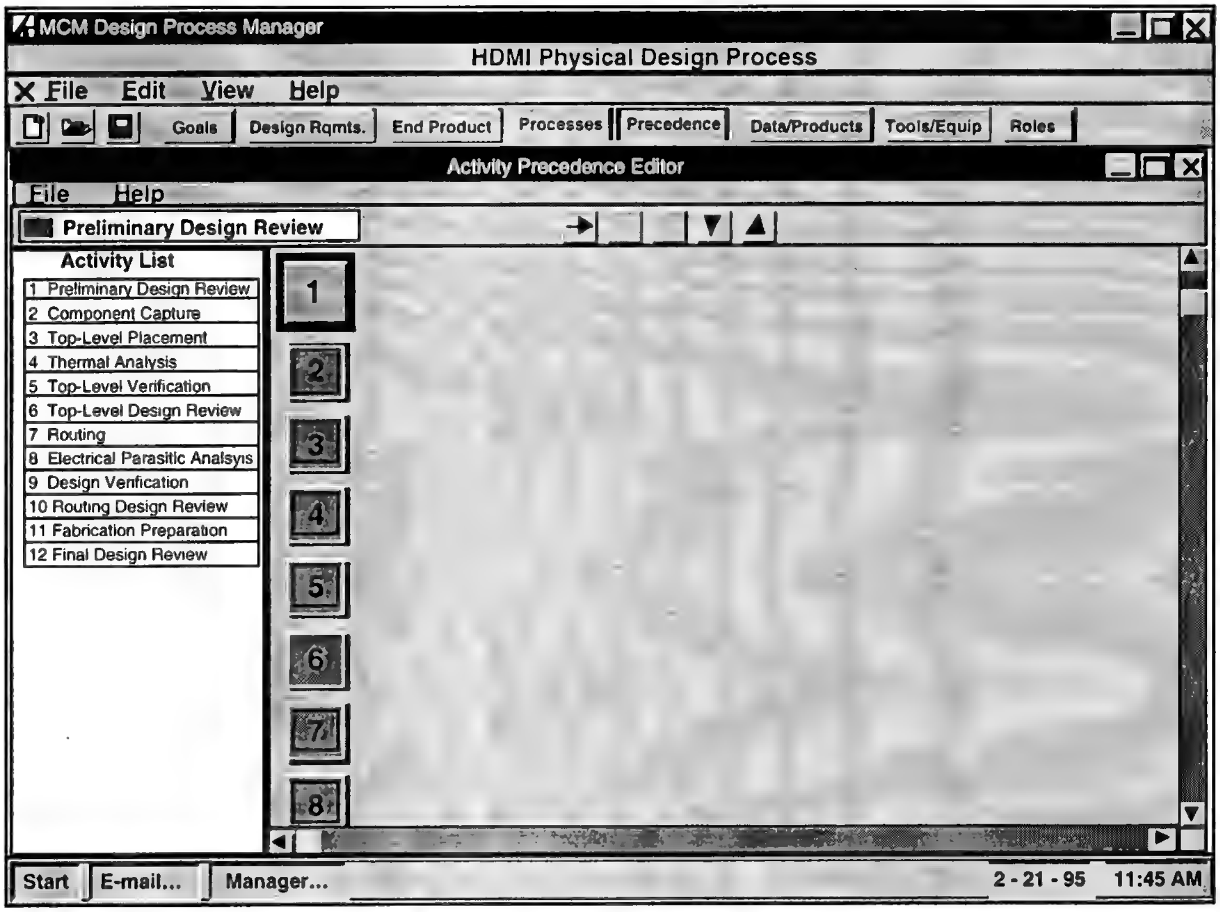


Figure 6-6. Initial View of Activity Precedence Editor Window

The process engineer then uses the editing facilities of the window to graphically specify the functional relationships between the activities (Figure 6-7). This is done at every level of the hierarchy.

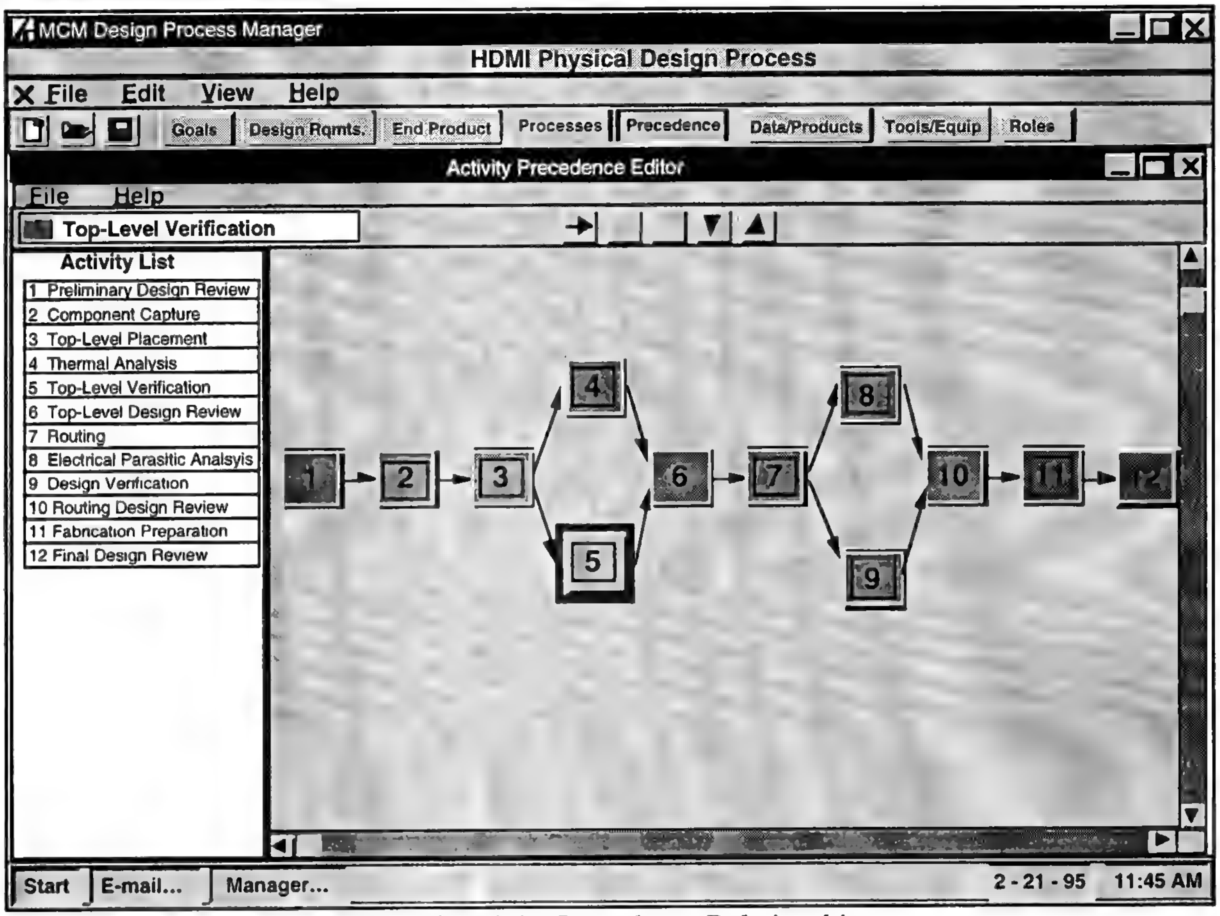


Figure 6-7. Graphical Depiction of Activity Precedence Relationships

Activity Browsing/Editing. The process engineer links each activity created to other elements of the IPPR as applicable. Activities to be performed by the designer during the execution of the design process are associated to specific goals, design requirements, tools/equipment, and roles. In addition, each activity has **entry criteria** composed of data/product inputs and pre-conditions, and **exit criteria** composed of data/product outputs and post conditions. Figure 6-8 shows the window created when an activity in the precedence graph is selected. This window is where the process engineer provides the information that links the activity to other elements of the IPPR, and ultimately where the designer browses/reviews the information about an activity that he/she is going to perform.

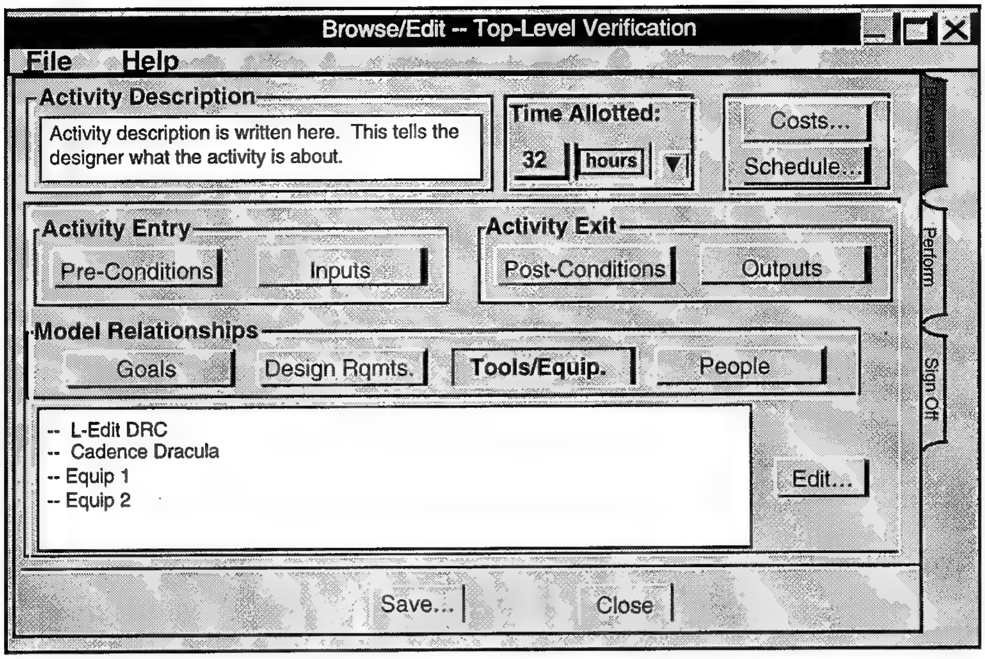


Figure 6-8. Activity Browse/Edit Window

The process engineer can also print a report that summarizes, for each activity, the relationships between an activity and other elements of the model as specified (Figure 6-9).

Activity Information for HDMI Physical Design Process Model	
Activity: Top-Level Placement	
Goals:	Satisfy Manufacturing Assembly Requirements
Design Requirements:	Electrical Performance Thermal Performance Size Weight Speed
End Product Components:	Device Location
Data Items/Interim Products:	Top-Level Plots and Part Replacement
Tools/Equipment:	L-Edit L-Edit SPR MGC Layout
Roles:	Designer
Variables/Conditions:	Activity 3 = *complete*
Schedule:	3/27/95 to 4/5/95 5 days
Costs:	$3,570.00

Figure 6-9. Format of Activity Information Report

Static and Statistical Model Analysis. In addition to providing editing capabilities, the MCM-DPM provides both static and statistical analysis capabilities to the process engineer. The statistical information (Figure 6-10) gives the process engineer information about the model elements, precedence graph structure, and activity-related numbers.

Statistical Information About HDMI Physical Design Process Model

Total Model Elements	
Total Number of Goals/Subgoals:	12
Total Number of Design Requirements:	16
Total Number of End Product Components:	17
Total Number of Process Activities:	26
Total Number of Data Items/Interim Products:	15
Total Number of Tools/Equipment:	32
Total Number of People:	5
Total Number of Variables/Conditions:	12
Total Number of Elements:	**135**

Precedence Graph Structure	
Number of Decomposition Levels:	3
Number of Activities without Subactivities:	3
Number of Activities without Predecessors:	0
Number of Activities without Successors:	0

For Each Activity:

Type	Max Number	Min Number	Avg Number
Goals	4	1	2.50
Design Requirements	6	1	3.50
End Product Components	4	0	2.13
Data Items/Interim Products	12	1	7.25
Tools/Equipment	4	1	2.50
People	2	1	1.50
Variables/Conditions	8	0	1.50

Figure 6-10. Sample Statistical Information Report

The MCM-DPM performs a static analysis of the IPPR along several dimensions. Figure 6-11 shows some of the information produced in this report to aid the process engineer in determining the correctness and completeness of the model.

Static Analysis About HDMI Physical Design Process Model

Activities Missing Some Information	
Activities without Goals:	<Activity Name>
Activities without Design Requirements:	<Activity Name>
Activities without End Product Components:	<Activity Name>
Activities without Data Items/Interim Products:	<Activity Name>
Activities without Tools/Equipment:	<Activity Name>
Activities without Roles:	<Activity Name>
Activities without Variables/Conditions:	<Activity Name>
Model Elements Not Asociated With Any Activity	
Goals Not Included in any Activity:	<Goal Name>
Design Requirements Not Included in any Activity:	<Design Requirement Name>
End Product Components Not Included in any Activity:	<End Product Component Name>
Data Items/Interim Products Not Included in any Activity:	<Data Item/Interim Product Name>
Tools/Equipment Not Included in any Activity:	<Tools/Equipment Name>
Roles Not Included in any Activity:	<Role Name>
Variables/Conditions Not Included in any Activity:	<Variable/Condition Name>
Precedence Graph Structure Problems	
Illegal Precedence Relation:	None Found
Illegal Iteration and Branch Pairs:	None Found
Endless Precedence Cycle:	None Found

Figure 6-11. Format of Static Information Report

Input/Output Transformations. The MCM-DPM maintains a hierarchy of "Stores", i.e., locations for each data item, interim product, end product component, software tool, and equipment. The process engineer uses the capabilities shown in Figure 6-12 to specify the relationships between activity entry elements and activity exit elements. These relationships include updating, "absorbing", and using the elements during the performance of an activity. This information allows the MCM-DPM to track the life cycle of every design object during the design process.

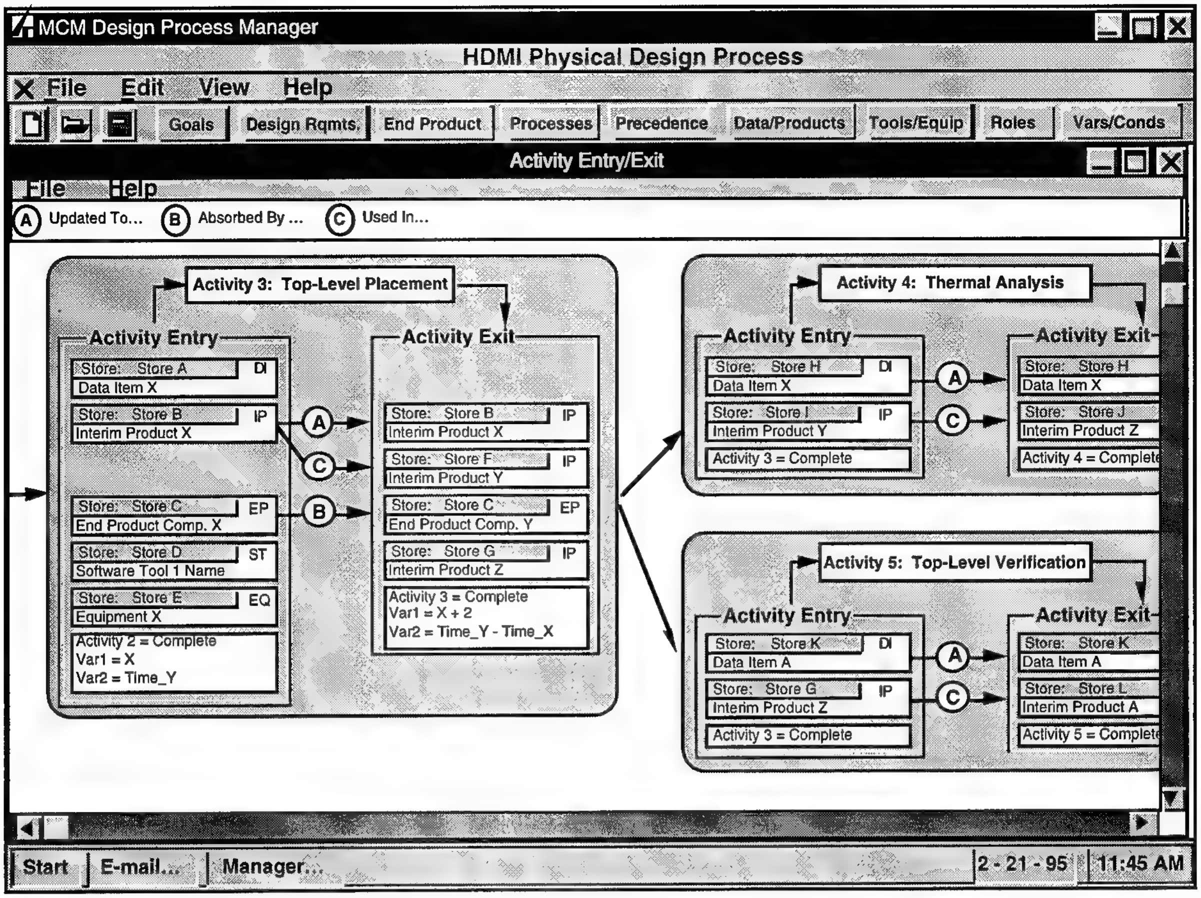


Figure 6-12. Activity Entry/Exit Window

When the process engineer has finished creating/tailoring the IPPR, the MCM-DPM system is ready to support the manager and designers in executing the MCM design process. In the execution mode, the manager specifies the project schedule and cost associated with each activity. The manager also assigns each task to the design team members (i.e., roles). When this is complete, the MCM-DPM guides the team members (i.e., designers) through their individual tasks. Additional information about manager-system interactions are discussed in Section 6.4. The following section will present the interaction of the designer with the MCM-DPM.

6.3 Designer-System Interaction

This subsection presents the interactions between the designer and the MCM design process manager. The MCM-DPM usage concept is conveyed through a series of screens that follow. These screens depict how the designer might enter the system, evaluate his/her assignments, determine the course of action for a particular assignment, make any necessary modifications to the IPPR, perform the activities associated with his/her assignment, evaluate the results, and continue with the guidance on design objectives provided by the MCM-DPM. Figure 6-13 presents the designer's MCM-DPM usage concept.

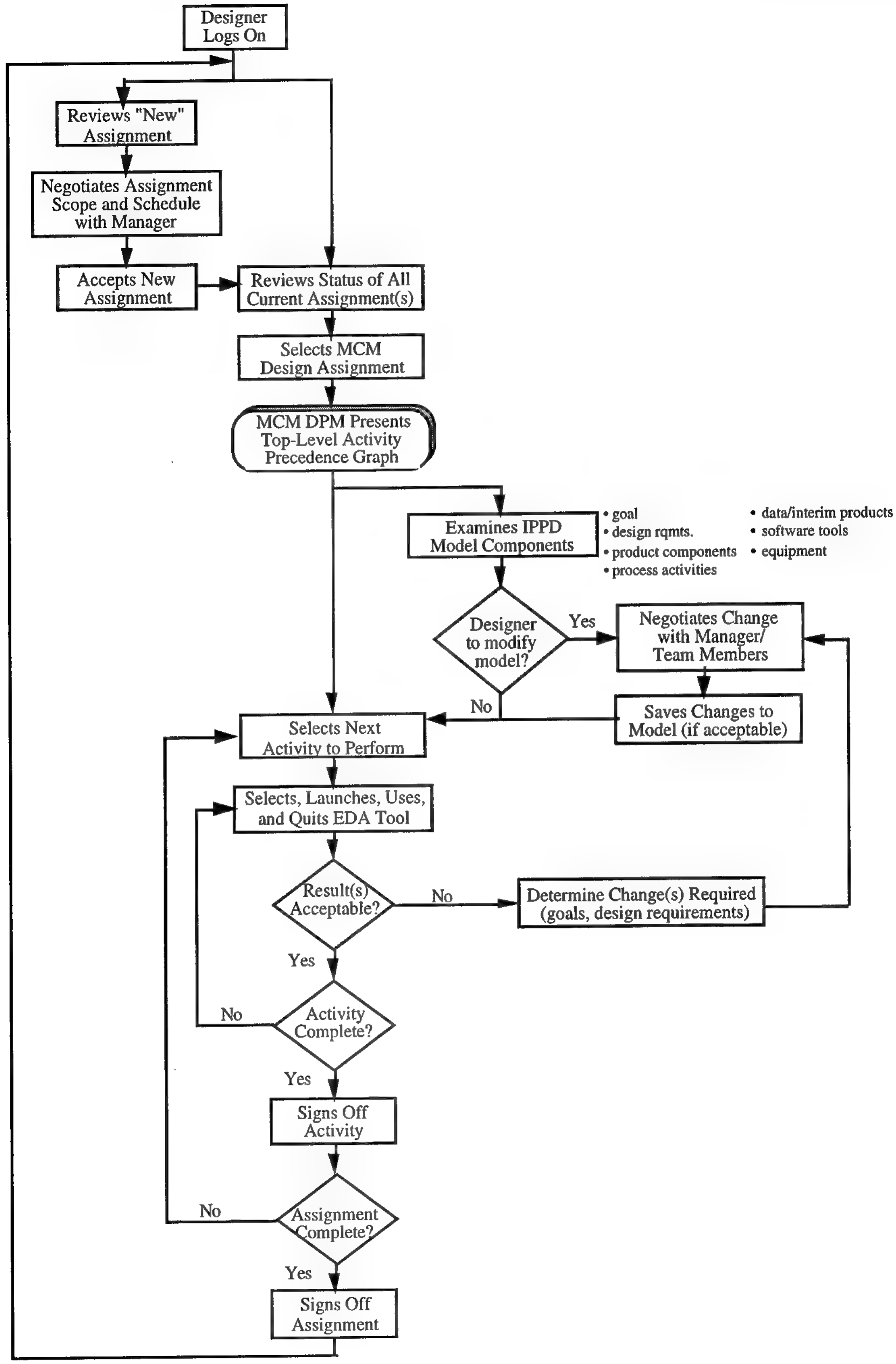


Figure 6-13. Designer-System Interaction Flow

Evaluating Designer Assignments. The designer enters the system and evaluates his/her assignments. In doing so, he may discover that a new assignment has been given to him. In that case he selects to review the new assignment (Figure 6-14). He may choose to negotiate changes of scope and schedule with the manager before accepting the assignment. He does this via the manager e-mail facilities.

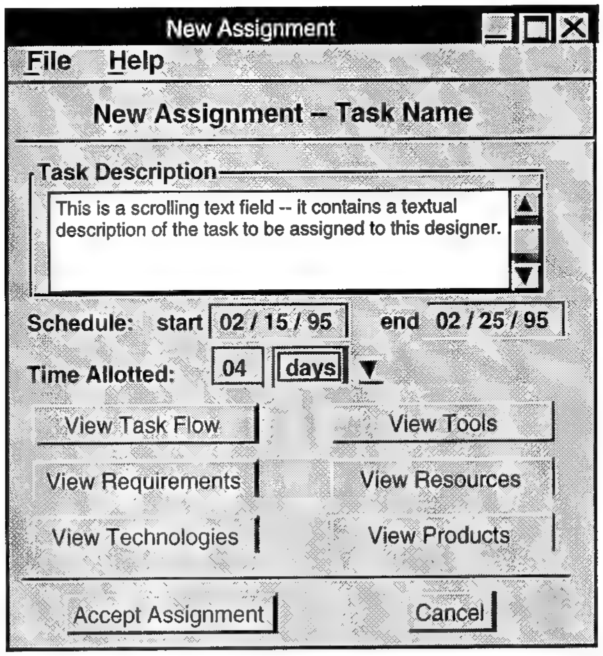


Figure 6-14. Reviewing New Assignment

After reviewing new assignments, if any, designers can review the status of all of current assignments. They can select the assignment they will work on at that time. The MCM-DPM presents the designer with the top-level activity precedence graph for that project.

Design Process Guidance. The Activity Selection window is a color-coded activity precedence graph. This graph guides the designer through the design process by identifying the "ready" activities assigned to him (Figure 6-15). The designer can simply select the next available activity and perform the required action(s).

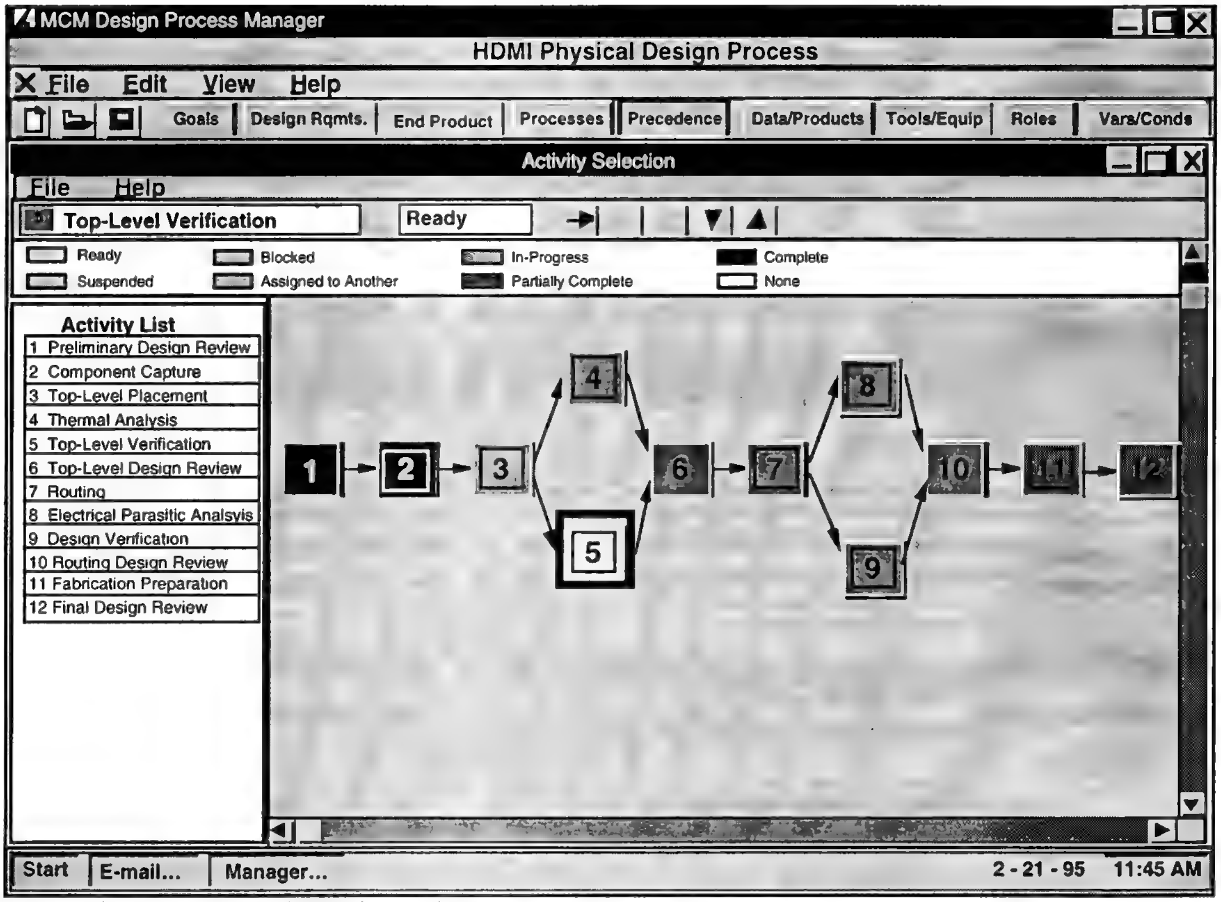


Figure 6-15. Activity Selection Window

Although the designer may select an activity from the Activity Precedence Graph display, he/she may decide to first examine the other components of the IPPR. For example, the designer may want to view the end product component hierarchy to gain a better understanding of the overall product to be created before attempting his assigned activities. Or in another example, the designer may have preliminary data from earlier attempts at the activity that cause him/her to introduce a change to the IPPR. He would utilize the e-mail facilities to communicate his suggestions to the other team members and, if approved, make the changes to the model.

Figure 6-16 shows the window that appears when the designer selects an activity to perform. This window shows the inputs, outputs, tools and equipment, and design modifiers associated with the selected activity.

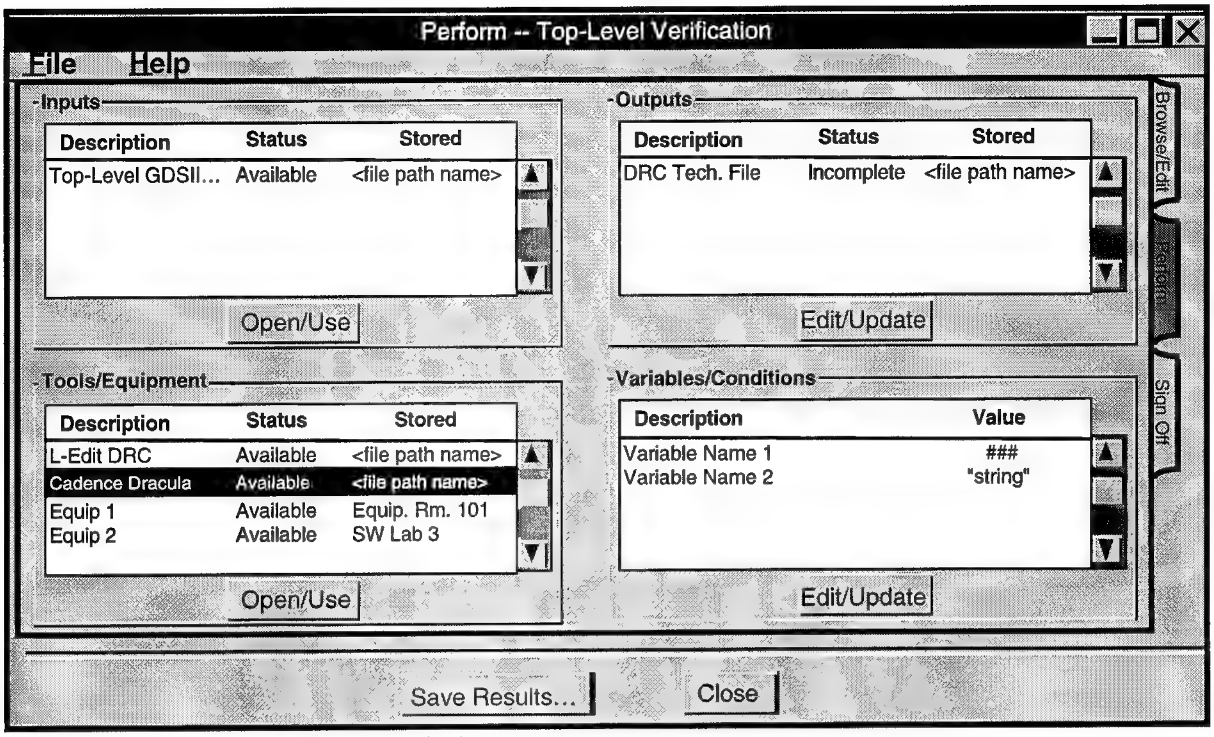


Figure 6-16. Perform Activity Window

To select an EDA software tool, the designer simply highlights it on the list of tools/equipment. The MCM-DPM opens the EDA tool for the designer and maintains the relevant information at the bottom of the window (Figure 6-17). To return to the MCM-DPM system, the user either quits the EDA tool or clicks on the MCM-DPM button at the bottom of the window.

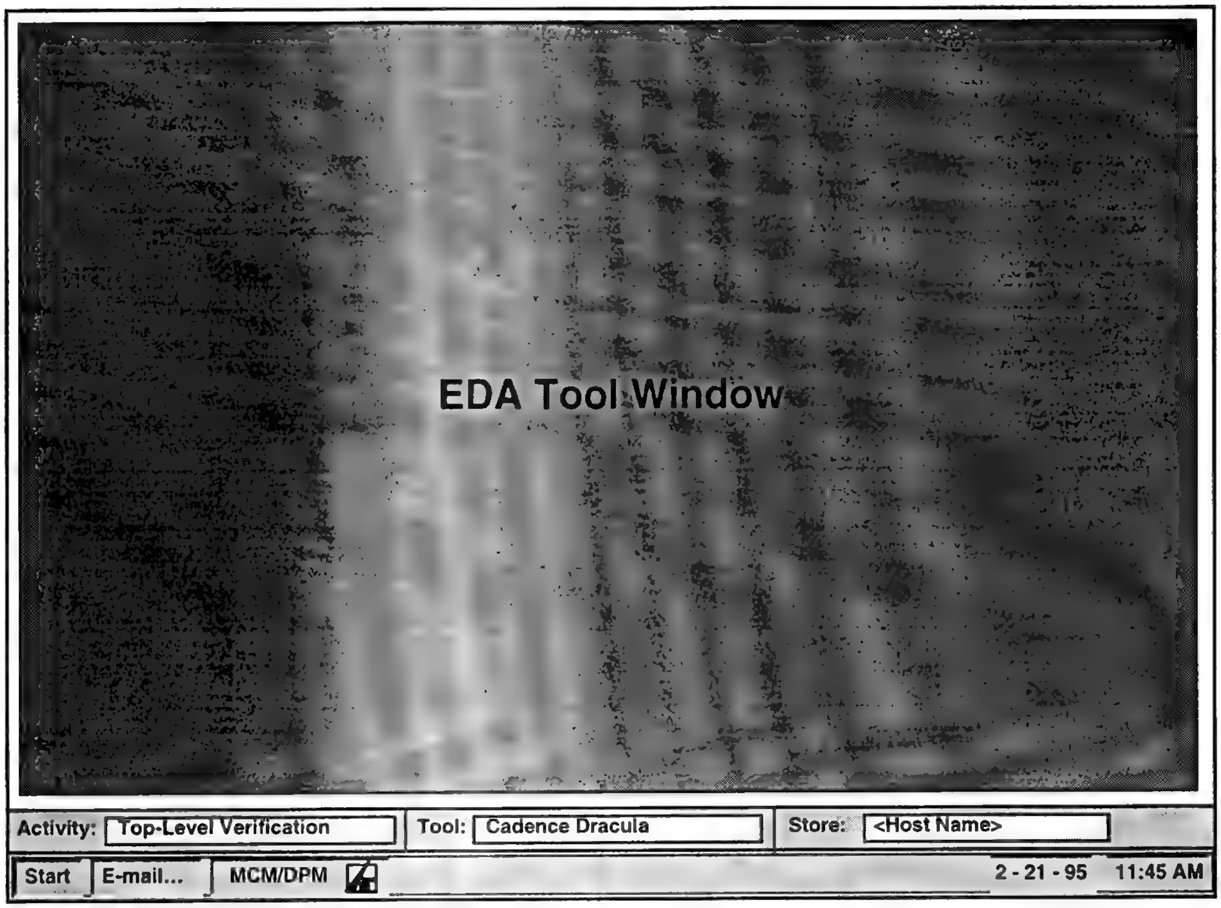


Figure 6-17. Launching EDA Software Tool

When the designer has completed the activity with acceptable results, he/she signs-off the activity using the window below (Figure 6-18) and selects the next available activity from the Activity Precedence Graph. However, if the assignment is also complete, he/she returns to reviewing the status of other assignments and begins the design process for the next project.

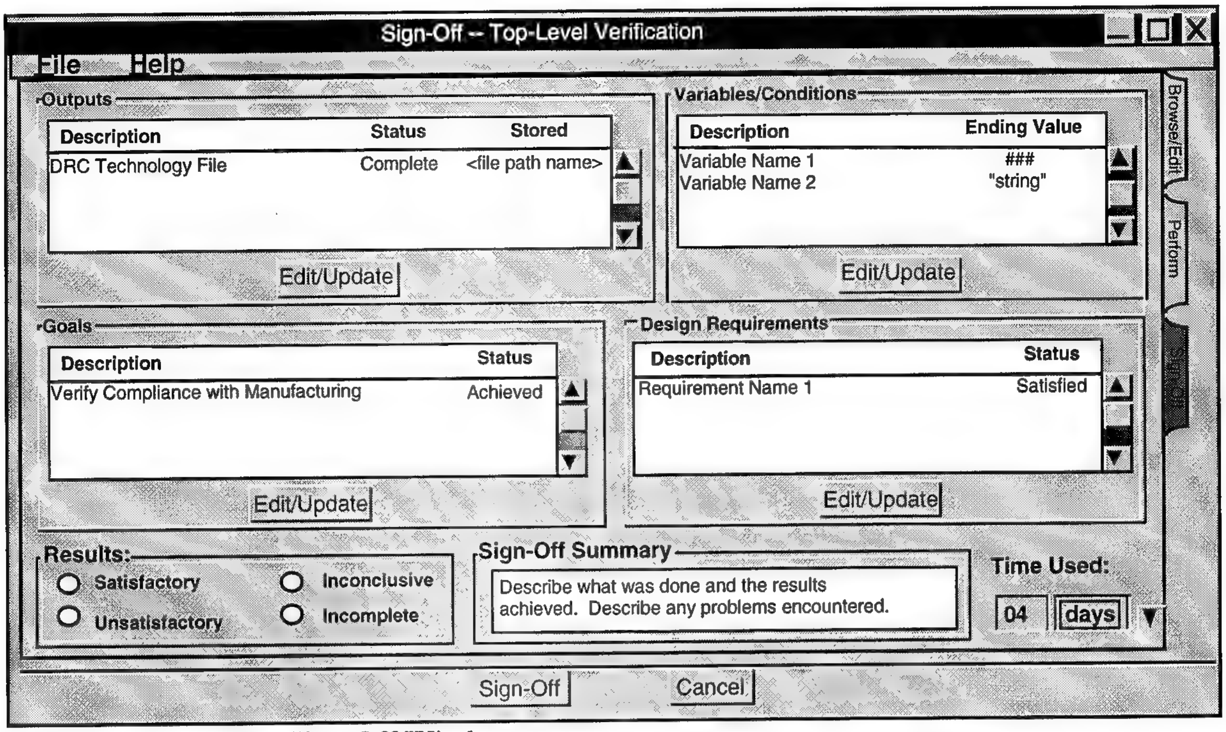
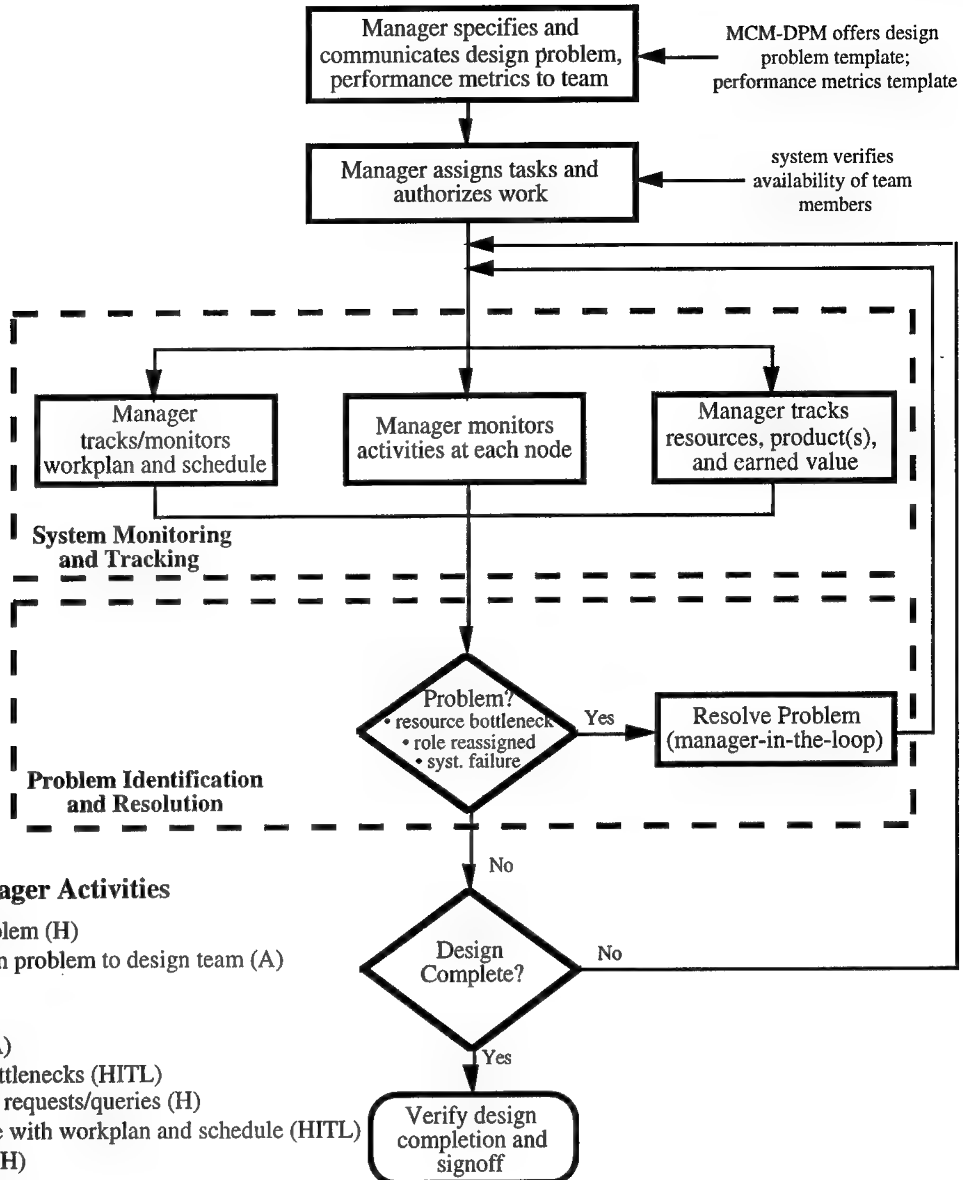


Figure 6-18. Activity Sign-Off Window

6.4 Manager-System Interaction

This subsection presents the functionality and decision support offered by the MCM-DPM for the design team manager. Specifically, it shows and discusses how the manager would utilize the scheduling and job costing functions of the MCM-DPM system to assign the various tasks created by the process engineer to the various members of the design team. It also describes how the manager can monitor the realtime status, trends, historical data, and statistical information of any design object in the IPPR. The manager also has access to the editing capabilities offered by the MCM-DPM in order to resolve problems during the execution process. The manager could modify elements of the IPPR to resolve resource bottlenecks, reassign roles, or circumvent system failures, for example. Figure 6-19 describes the manager's usage concept for the MCM-DPM.



Manager Activities

- specifies design problem (H)
- communicates design problem to design team (A)
- assigns tasks (H)
- authorizes work (H)
- monitors progress (A)
- resolves conflicts/bottlenecks (HITL)
- responds to designer requests/queries (H)
- monitors compliance with workplan and schedule (HITL)
- schedules meetings (H)
- approves work-in-progress (H)
- tracks earned value (A)
- anticipates and circumvents problems (HITL)

LEGEND	
H	: human activity, system supports with menus, forms, etc.
A	: automated activity
HITL	: human-in-the-loop

Figure 6-19. Manager-System Interaction Flow

Schedule, cost, and resource utilization information is provided by the manager to the MCM-DPM system through integration with project management facilities. Figures 6-20 through 6-22 show a sample project schedule chart, sample project timeline, and sample data input window which is used to enter schedule, cost, and resource information for each activity in the IPPR.

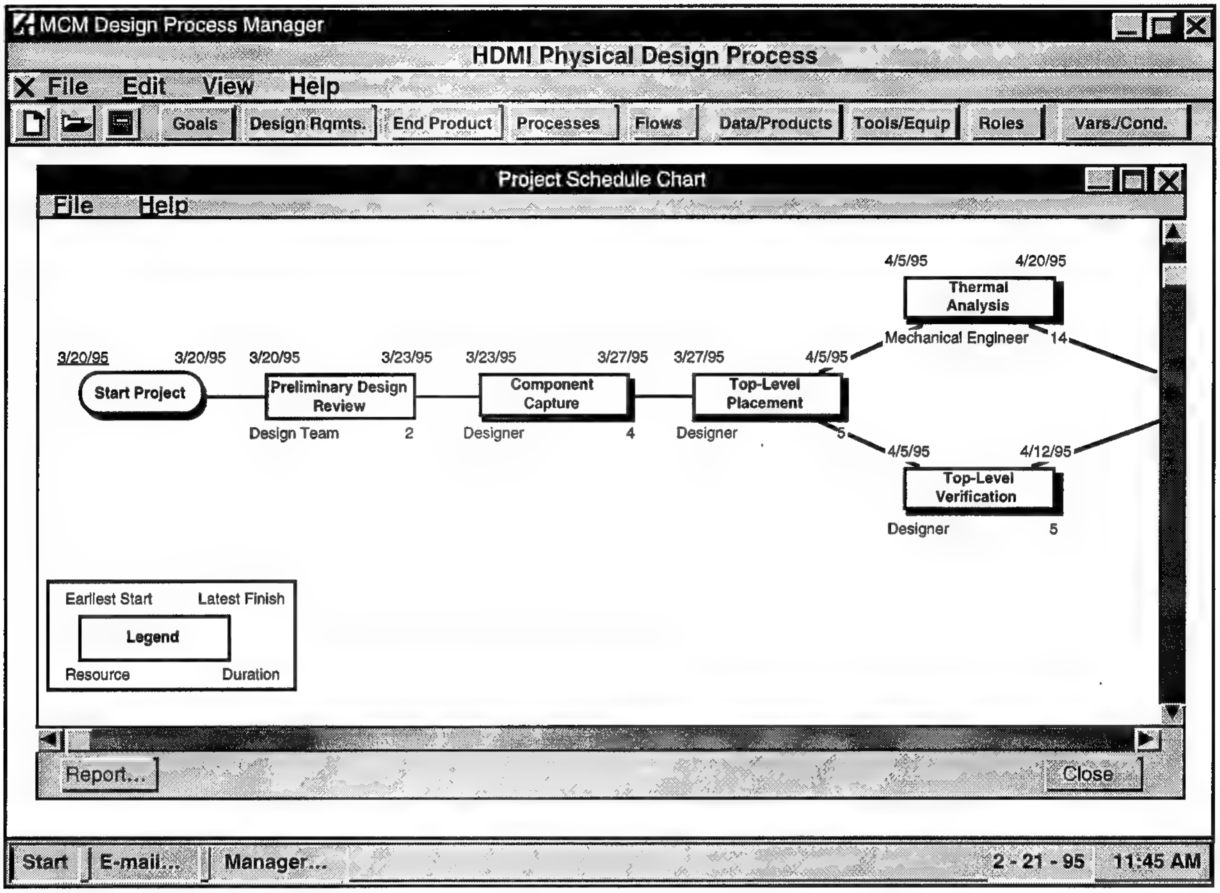


Figure 6-20. Sample Project Schedule Chart

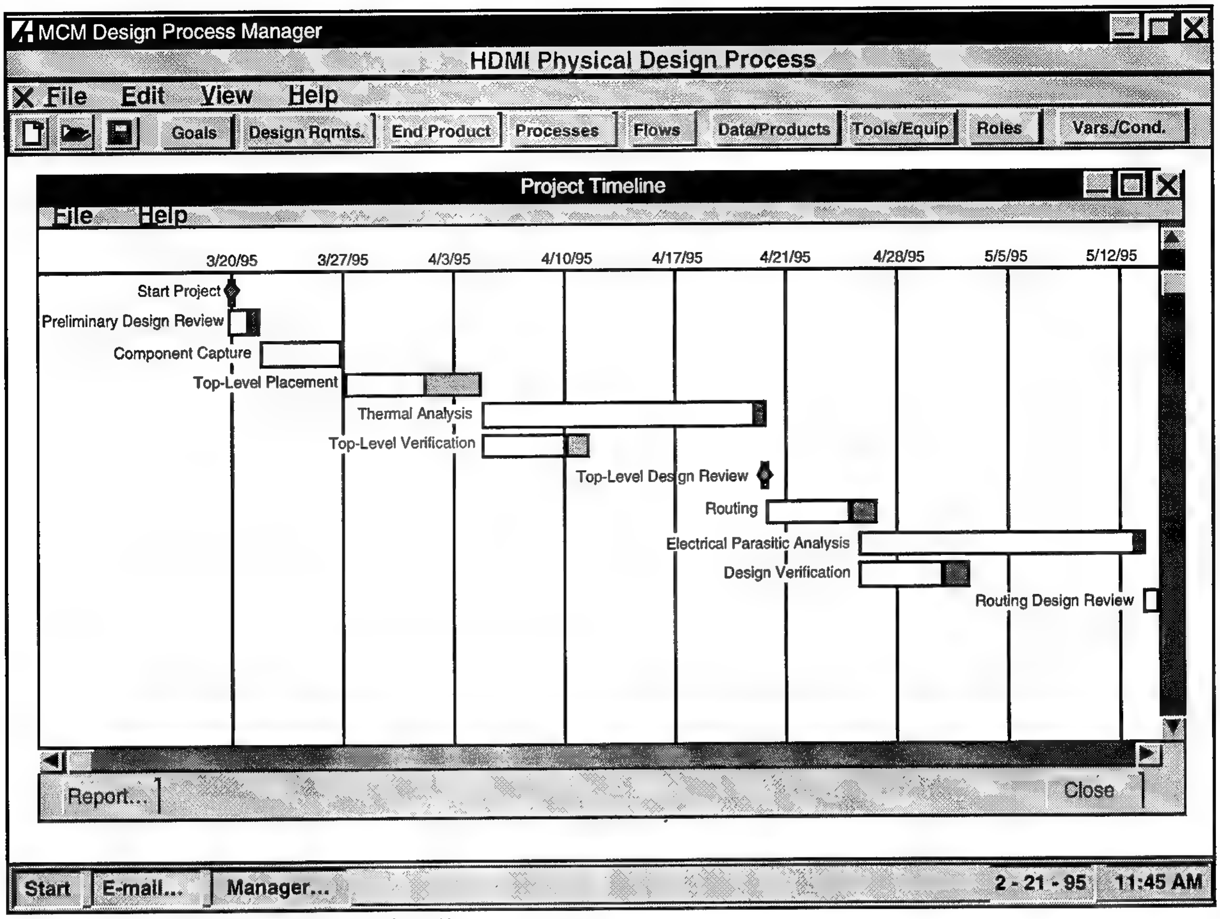


Figure 6-21. Sample Project Timeline

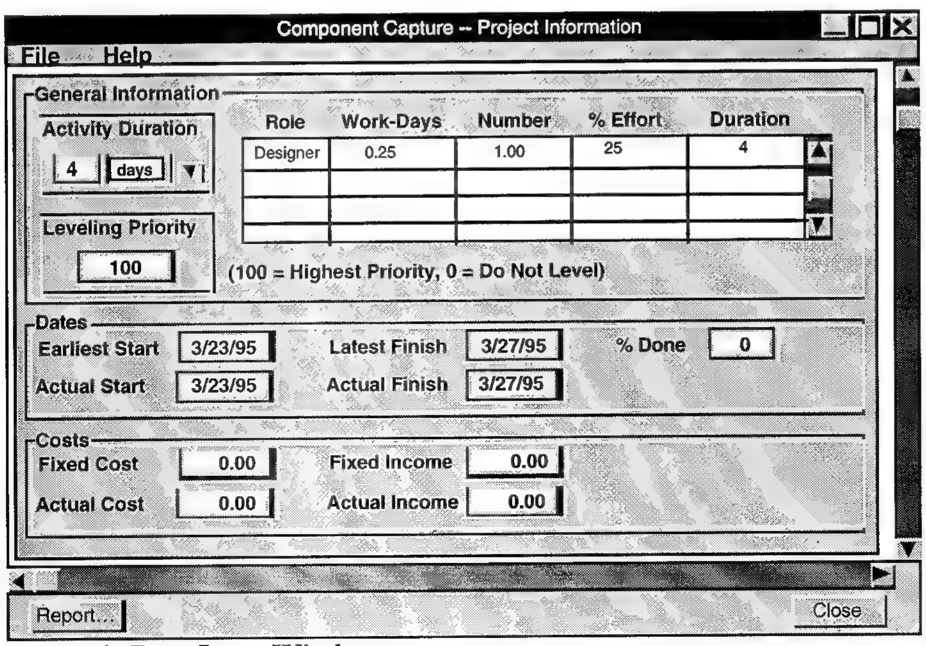


Figure 6-22. Sample Data Input Window

Figure 6-23 shows a sample color-coded window that allows the manager to view the current status of any activity in the IPPR. The manager can view a similar window for the goals hierarchy, the design requirements hierarchy, the data/interim products hierarchy, the tools/equipment hierarchy, the roles hierarchy, and the list of design modifiers.

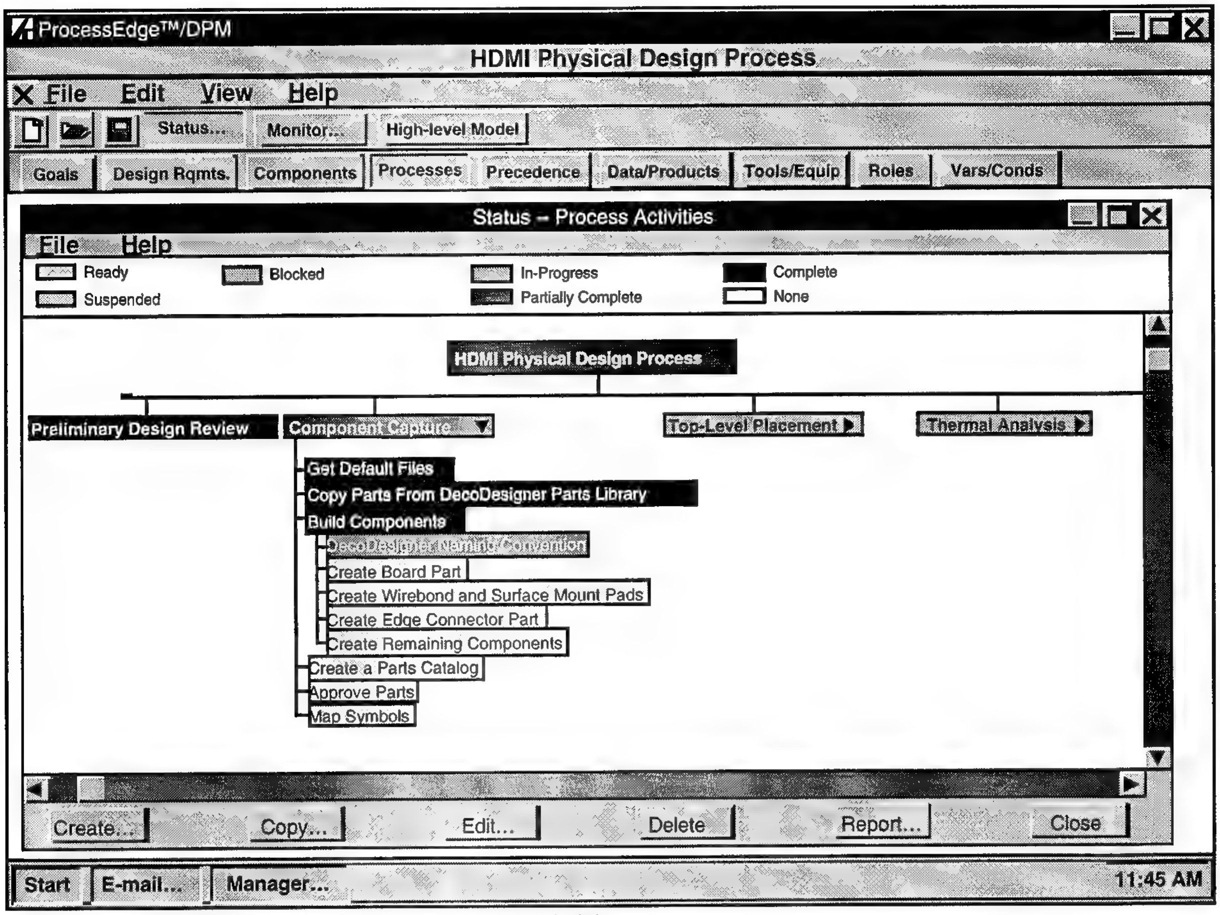


Figure 6-23. Monitor Status of Process Activities

Figures 6-24 through 6-26 show how the manager is able to monitor, in realtime, the status, performance/usage histograms, and statistics related to process activities, software tools, and human resources (roles).

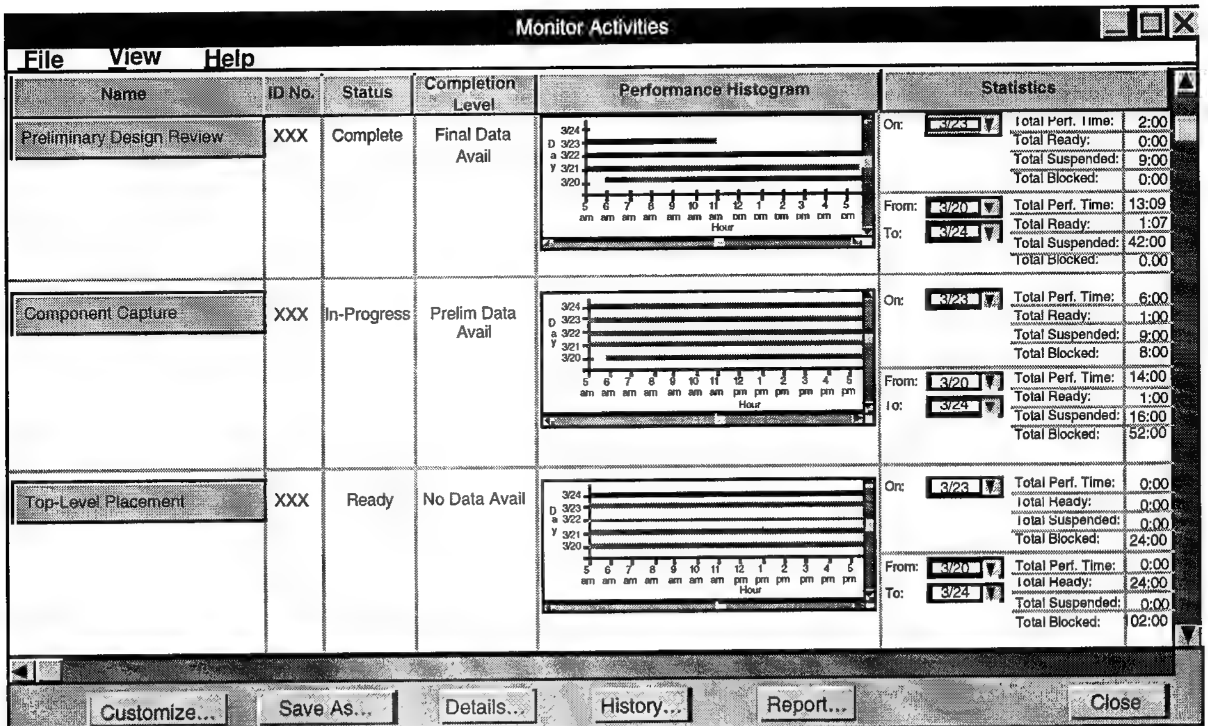


Figure 6-24. Monitor Activities Performance

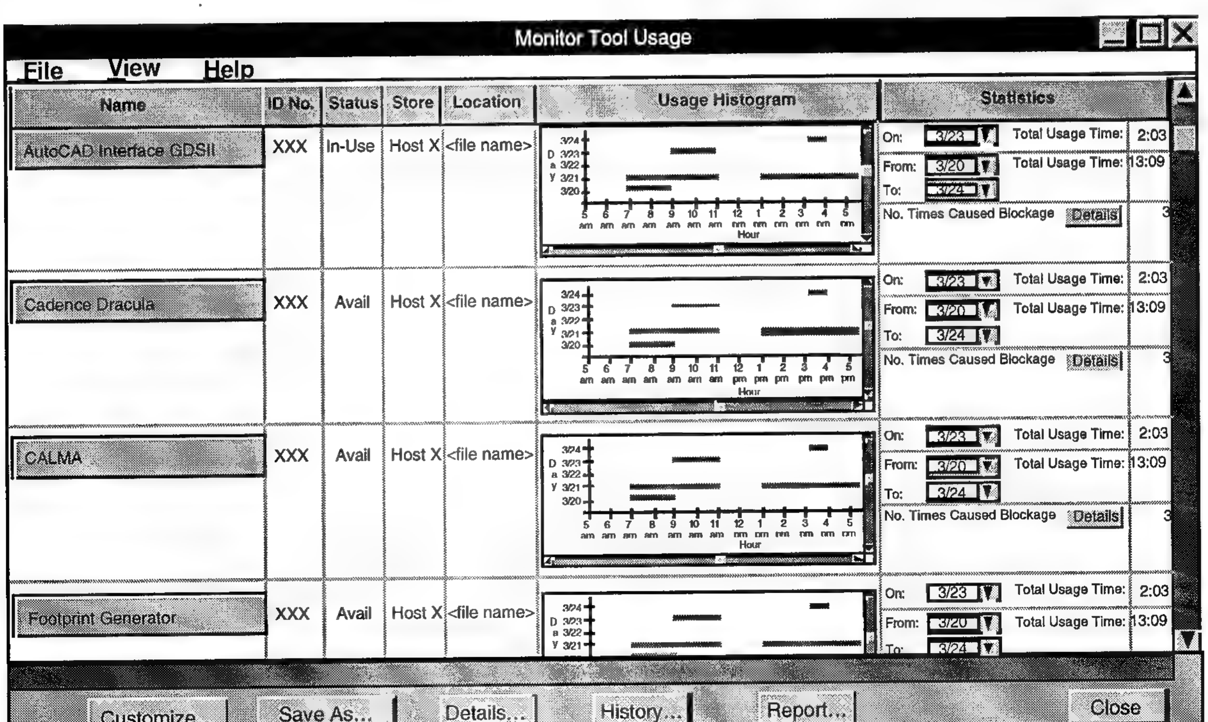


Figure 6-25. Monitor Software Tool Usage

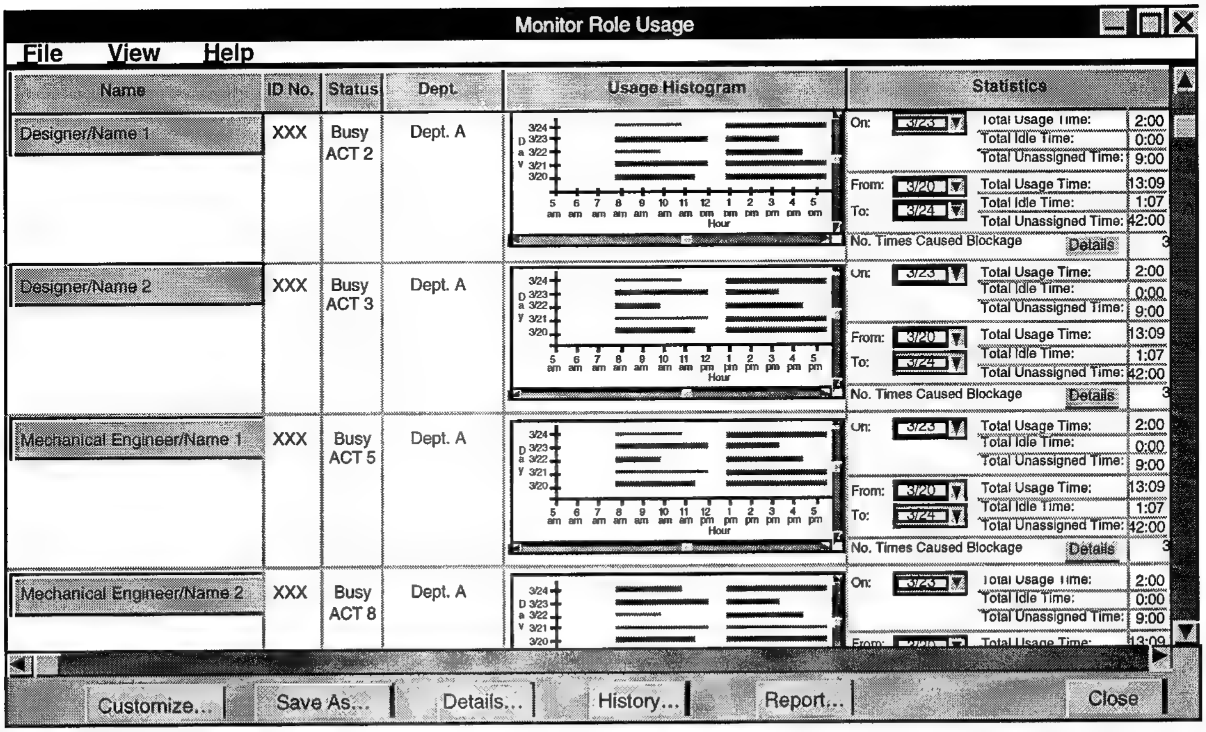


Figure 6-26. Monitor Role Usage

The MCM-DPM allows the manager to monitor the life cycle of any design object in the IPPR. Figure 6-27 shows the color-coded window that traces the transformations of an object between activities in the design process.

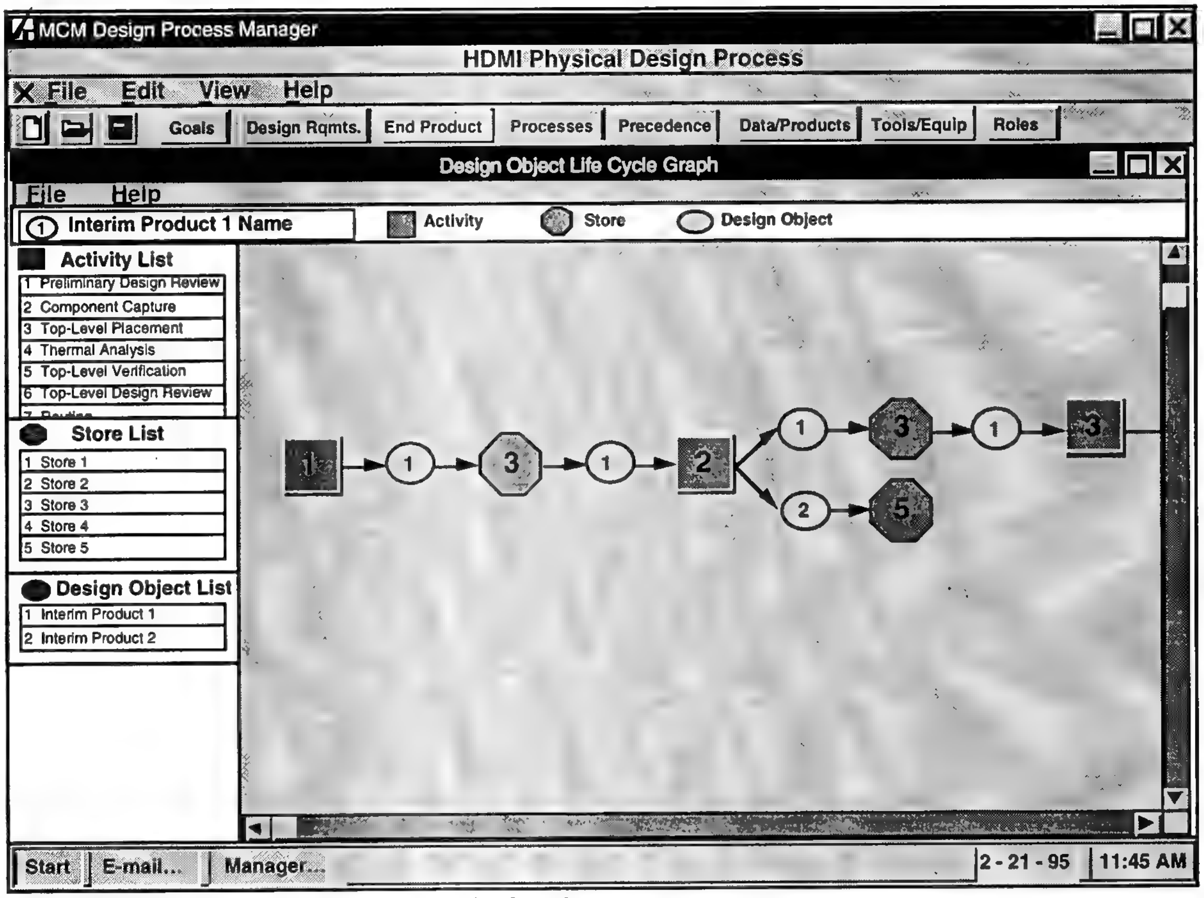


Figure 6-27. Design Object Life Cycle Graph

7. PHASE II IMPLEMENTATION PLAN

This section is divided into three major subsections. Subsection 7.1 presents a summary of tasks to be performed in Phase II. Subsection 7.2 provides a detailed discussion of each task in terms of inputs, approach, and outputs.

7.1 Task Summary

The Phase II work plan is organized into fifteen tasks that specifically respond to the objectives defined in the previous section. Figures 7-1(a) and 7-1(b) presents the work plan and the inter-relationships between the work tasks.

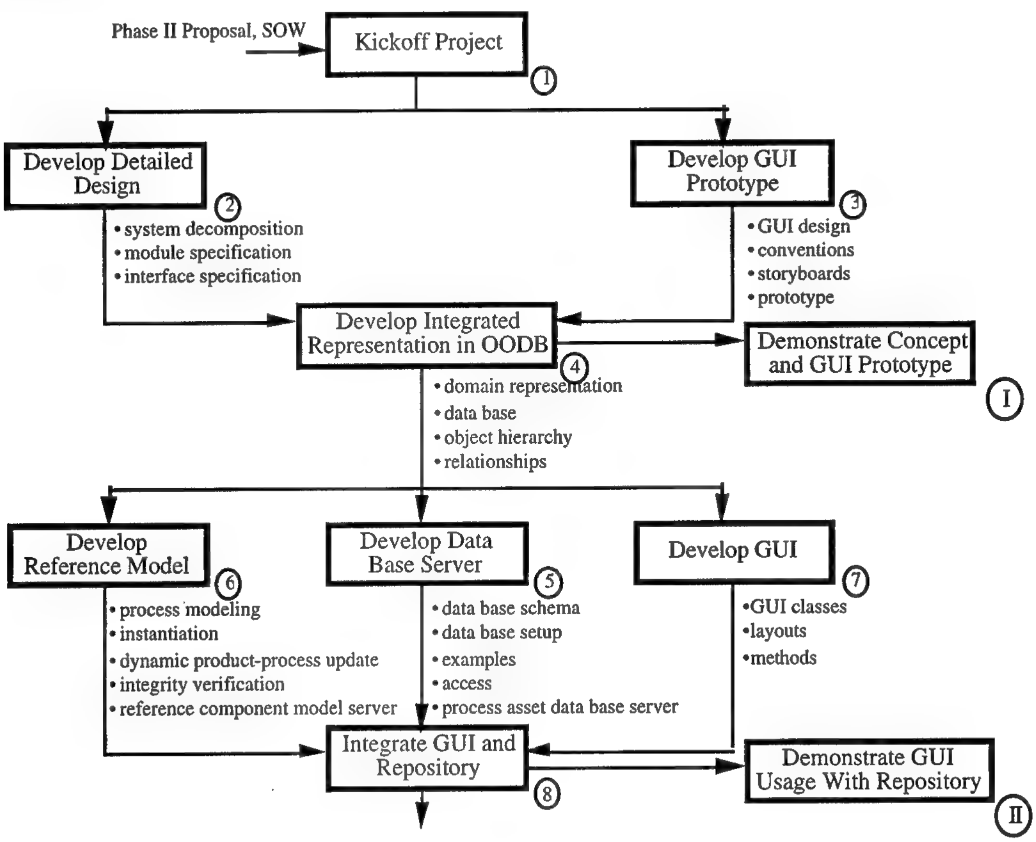


Figure 7-1(a). Phase II Work Plan (Tasks 1 through 8)

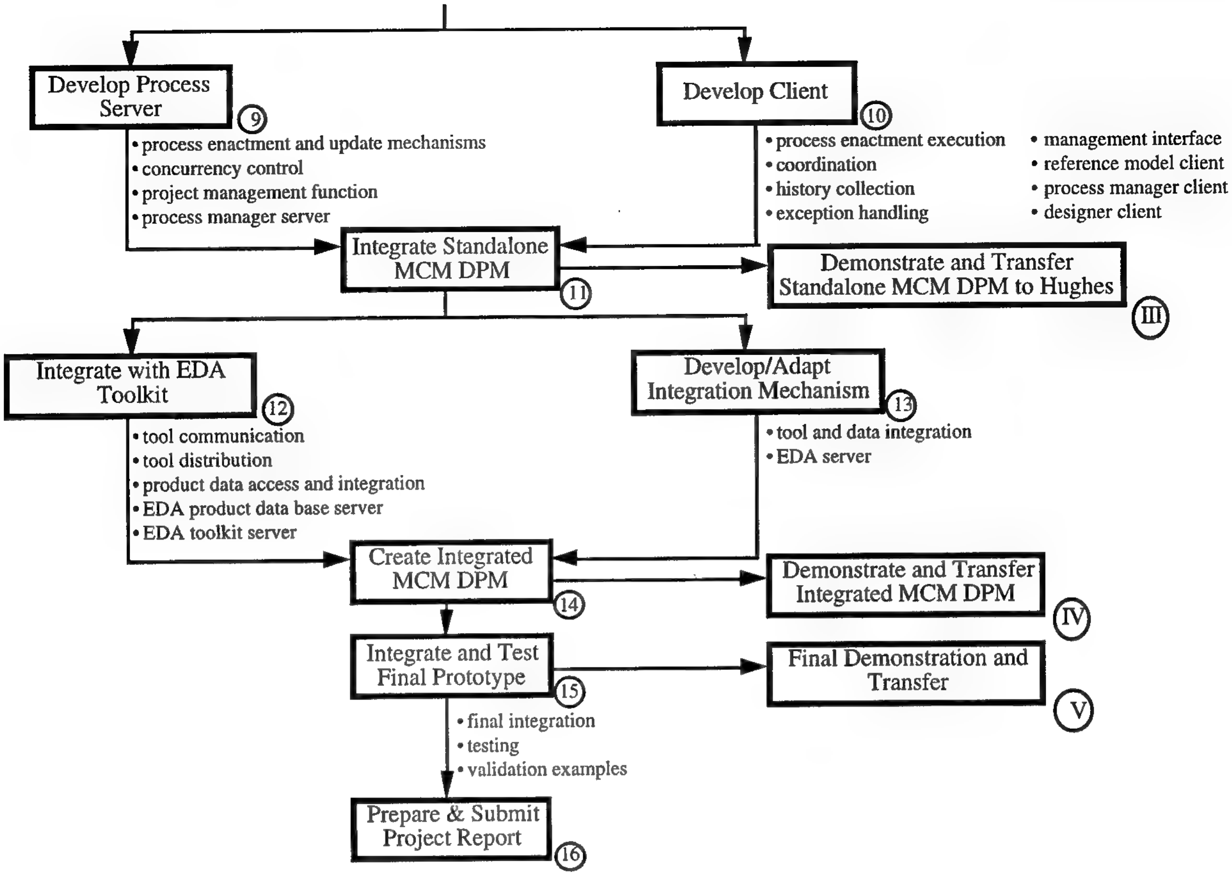


Figure 7-1(b). Phase II Work Plan Continued (Tasks 9 through 16)

7.2 Work Plan

This subsection provides details of the sixteen tasks and five demonstrations that lead to a robust implementation of the MCM Design Process Manager.

Task 1. Kickoff Project

The objective of this task is to review the proposed effort and make necessary adjustments to the schedule to make maximum impact on the ASEM program and MCM contractor needs. Specifically, we will present the MCM Design Process Manager System Concept and implementation in suitable detail. We will review our approach to process tailoring to meet the requirements of the various contractors. The product of this task will be a streamlined SOW directed to rapid product development with well-defined demonstration milestones and delivery schedule.

Task 2. Developed Detailed Design

Based on the architectural design completed in Phase I, this task will create a detailed system decomposition, module specification, and interface specification. The set of documents created during the performance of this task will describe the details of the software architecture and design of the MCM Design Process Manager.

Task 3. Develop User Interface Prototype

The purpose of this task is to design and prototype the Graphical User Interface (GUI) for the MCM Design Process Manager. During the performance of this task, the selected GUI development toolkit will be purchased and installed. The user interface conventions and the MCM Design Process Manager concept of operation will be storyboarded. Finally, a GUI prototype will be implemented and demonstrated.

Task 4. Develop Integrated Representation in the Selected Object-Oriented Data Base

The purpose of this task is to implement an integrated MCM product-process representation in terms of an OODB. The OODB implementation of the integrated MCM product-process representation include data base object classes and relationships that describe MCM design products and processes. The output of this task is the object hierarchy for the integrated MCM product and process representation. At the conclusion of this task, we will give our first demonstration.

Demonstration I. The purpose of this demonstration is to present the overall system concept, the object-oriented database, and the GUI prototype of the MCM Design Process Manager.

Task 5. Develop Data Base Server

This task will implement access and service functions of the data base server including data base setup, consistency and completeness checking manipulation, search, queries, examples, and access methods. There are two data base servers in the MCM Design Process Manager: the Process Asset Data Base Server which oversees access to the integrated MCM product-process representation, and the Project Asset Data Base Server which oversees access to the MCM product data bases.

Task 6. Develop Reference Process Model

This task will define and analyze a set of MCM reference process models and components in the OODB. This reference model is based on the information collected during Phase I. It will also implement the Reference Component Model Server with functions that enable process modeling, instantiation, dynamic product-process update, and integrity verification.

Task 7. Develop GUI

The objective of this task is to implement the GUI for the MCM Design Process Manager, including GUI classes, layouts, and methods.

Task 8. Integrate GUI and Repository

The objective of this task is to integrate the GUI and repository created in previous tasks. At the conclusion of this task we will give a demonstration of the GUI and the repository functionalities.

Demonstration II. The purpose of this demonstration is to show the key features of the repository for the integrated MCM product-process representation and GUI for the MCM Design Process Manager.

Task 9. Develop Process Server

This task will implement the Process Server of the MCM Design Process Manager including process execution engine (the development strategy will be chosen during detailed design), process update mechanism, concurrency control, and project management functions.

Task 10. Develop Client Software
This task will implement three different MCM Design Process Manager clients (Reference Model Client, Process Manager Client, and Designer Client) for process execution, coordination, history collection, exception handling, and management interface The Reference Model Client is responsible for creating process instances to be executed. The Process Manager Client is a management interface for monitoring, controlling, and managing MCM design activities. The Designer Client is a development interface for MCM designers to perform MCM design activities.

Task 11. Integrate Standalone MCM Design Process Manager (DPM)
The objective of this task is to create a "standalone" MCM Design Process Manager, i.e., the software will not be integrated with EDA toolkits. This software will be demonstrated to end users and sponsor for their evaluation and feedback.

Task 12. Integration Software with EDA Toolkit
The objective of this task is to interface the MCM Design Process Manager software to the selected EDA toolkit. Specifically, we will establish mechanisms for tool communication, tool distribution, product data access and integration, EDA Product Data Base Server, and EDA Toolkit Server.

Demonstration III. The purpose of this demonstration is to show end users and sponsors the major MCM Design Process Manager functionalities in a standalone mode.

Task 13. Develop Integration Mechanism
The purpose of this task is to implement the tool and data integration mechanisms for the MCM Design Process Manager embodied in the EDA Server.

Task 14. Create Integrated MCM Design Process Manager
The objective of this task is to integrate the process management, the EDA toolkit, and the integration mechanism within a fully functional MCM Design Process Manager. At the conclusion of this task, we will be ready to demonstrate the operation of the integrated MCM Design Process Manager.

Demonstration IV. The purpose of this demonstration is to showcase the capabilities of the integrated MCM Design Process Manager working with the selected EDA toolkit.

Task 15. Integrate and Test the MCM Design Process Manager
The purpose of this task is to perform final integration, testing, and validation with example scenarios and data in order to make the MCM Design Process Manager into a workable environment. At the conclusion of this task, we will give a final demonstration of the total system with real examples and transfer the working software to Hughes, Newport Beach EDA environment.

Demonstration V. The purpose of this demonstration is to show the complete implementation and full range of capabilities of the MCM Design Process Manager with realworld MCM design examples.

Task 16. Prepare and Submit Project Report

This task will revise and finalize the development documents that were initially created during the earlier development activities. The set of documents includes: system requirements specification; detailed design specification; system modules and interface specification, test plan, guide, and history; system installation guide; and user's guide. The report will also provide a detailed commercialization plan including product positioning strategy, alliances with EDA vendors, marketing, distribution, and sales strategies, and sample brochures.

Task 17. Present Progress at ASEM and EP&I Conference

This task is concerned with presenting the work-in-progress and demonstrating the software functionality of the MCM Design Process Manager to the ASEM community and to participants in the Electronic Packaging and Interconnect Conference. These two conferences will provide a forum to showcased our evolving product and its capabilities to a fair segment of the potential customer base. Their review and feedback will help us in creating a more responsive product for their needs.

8. CONCLUSIONS AND PHASE II PLANS

In this Phase I Final Technical Report, we have presented the overall system concept, key tradeoffs, architecture, system components, and functionalities of the MCM Design Process Manager (MCM-DPM). We have also developed a series of computer screens based on Windows '95 "look and feel" to convey the key functionalities and features of the prototype for user and sponsor evaluation and feedback. For transparent communication, remote object invocation, and distributed object management we have selected XShell, a Distributed Object Management Environment (DOME) that is supposed to have a CORBA 2.0-compliant version by May 1995. For our repository, we have selected ObjectStore, an object-oriented repository from Object Design. XShell and ObjectStore have been successfully integrated and can be bought as an integrated package. This combination provides us with the capability for low-level process execution with persistent storage of state information. Nevertheless, the C^{++} software development effort with this approach is significant in creating the design process management layer on top of XShell. We have worked out the particulars of this development effort with XShell developers, who will work with us as necessary during the integration phase. Our target host environment is Hughes Aircraft Company, Newport Beach, CA. Their MCM design environment consists of their DecoDesigner toolkit as well as the Tanner Toolkit. Our solution will be compatible with either toolkit.

Our Phase II proposal provides a detailed implementation plan, key demonstration and integration milestones, and transition plan. Phase II will conclude with an operational prototype demonstrable within the Hughes environment. One of our key goals is to maintain the price of our client software around $1,000 to $1,200. To achieve this goal we will be working out a reduction in runtime license with Expersoft (and Object Design). We have already started talks on this subject. Expersoft has indicated their willingness to reduce their current runtime license fees from $500 to a lower figure.

REFERENCES

Brockman, J.B., Cobourn, T.F., Jacome, M.F., and Director, S.W. The Odyssey CAD Framework. IEEE DATC Newsletter on Design Automation, Spring 1992.

Brockman, J.B. and Director, S.W. A Schema-based Approach to CAD Task Management. Proceedings of the Third IFIP WG 10.2 Workshop on Electronic Design Automation Frameworks, Edited by T. Rhyne and F.J. Rammig, Elsevier Science Publishers, 1992.

Brockman, J.B. and Director, S.W. The Hercules CAD Task Management System. Proceedings of the International Conference on Computer-Aided Design, IEEE, 1991, pp. 254-257.

Brockman, J.B. and Director S.W. "The Hercules CAD Task Management System." Proceedings off the IEEE International Conference on Computer-aided Design, IEEE, 1991.

Bryant, R.E., Beatty, D., Brace, K. Cho, K. and Sheffler, T. COSMOS: A Compiled Simulator for MOS Circuits. Proceedings of the 24th ACM/IEEE Design Automation Conference, ACM, 1987, pp. 9-16.

Buckley, K.F. "Engineering Process Management System Springboard for CE." Proceedings of CE and CALS Exposition, June 1993.

Casotto, A., Newton, A.R., and Sangiovanni-Vincentelli, A. Design Management based on Design Traces. Proceedings of the 27th ACM/IEEE Design Automation Conference, ACM, 1990, pp. 136-141.

Chiueh, T. and Katz, R. A History Model for Managing The VLSI Design Process. Proceedings of the International Conference on Computer-Aided Design, IEEE, 1990, pp. 358-361.

Curtis, B., Kellner, M.I., and Over, J. in Process Modeling Communications of the ACM, vol, 35, no. 9, pp. 75-90, September, 1992.

Jacome, M.F. and Director, S.W. "Design Process Management for CAD Frameworks." 29th ACM/IEEE Design Automation Conference, pp. 500-505, 1992.

Jacome, M.F. and Director, S.W. Design Process Management for CAD Frameworks. Proceedings of the 29th ACM/IEEE Design Automation Conference, IEEE Computer Society Press, 1992, pp. 500-505.

Knapp, D.W. and Parker. "A Design Utility Manager: The ADAM Planning Engine." Proceedings of the 23rd ACM/IEEE Design Automation Conference, ACM Press, pp. 48-54, 1989.

Liebisch, D.C. and Jain, A. JESSI Common Framework Design Management -- The Means to Configuration and Execution of the Design Process. Proceedings of First European Design Automation Conference, GI/ACM/IEEE/IFIP, 1992, pp. 552-557.

Madni, A.M. A Scalable, Customizable MCM Design Process Manager. Intelligent Systems Technology, Inc. Quarterly Progress Report, SBIR Phase I, Contract No. DAAH01-94-C-R291, December 5, 1994.

Madni, A.M. A Conceptual Framework and Enabling Technologies for Computer-aided Concurrent Engineering (CACE). Plenary address & Invited Paper, Second International Conference on Human Aspects of Advanced Manufacturing and Hybrid Automation, August 12-16, 1990(b).

Madni, A.M. HUMANE: A Knowledge-Based Simulation Environment for Human-Machine Function Allocation. Proceedings of IEEE National Aerospace & Electronics Conference, Dayton, Ohio, May 1988.

Rumsey, M. and Farquhar, C. Unifying Tool, Data and Process Flow Management. Proceedings of First European Design Automation Conference, GI/ACM/IEEE/IFIP, 1992, pp. 500-505.

ten Bosch, K.O., Bingley, P., and van der Wolf, P. Design Flow Management in the NELSIS CAD Framework. Proceedings of the 28th ACM/IEEE Design Automation Conference, ACM, 1991, pp. 711-716.

van den Hamer, P. and Treffers, M.A. A Data Flow Based Architecture for CAD Frameworks. Proceedings of the International Conference on Computer-Aided Design, IEEE, 1990, pp. 482-485.

van der Wolf, P., Sloof, G.W., Bingley, P., Dewilde, P. Meta Data Management in the NELSIS CAD Framework. Proceedings of the 27th ACM/IEEE Design Automation Conference, ACM, 1990, pp. 142-145.

APPENDIX A: EDA STATUS AND TRENDS

EDA Integration and Interoperability Status

The CAD Framework Initiative (CFI) is an international cooperative effort within the electronic industry to define standard system interfaces and services that facilitate integration of design automation tools and design data. CFI provides an effective alternative to proprietary solutions. It also provides a vehicle for organizations to collectively invest in standard solutions that serve their individual tool-integration problems. CFI 1.0 standards, first piloted in 1992, are finally being certified and offered commercially. Initial CFI efforts have focused primarily on Design Representation and ASIC Logic Design. CFI is just beginning to look at process management but standards in this area are not expected for quite some time.

Today both users and EDA vendors recognize a growing "technology gap" between EDA capabilities and submicron process/fabrication demands. Despite the interoperability problems, users continue to buy the "best" individual tools and worry interoperability later. This is not surprising given that the cost of fixing interoperability failures is much less than the cost of missing a market opportunity.

Standards are having a mixed reception. While standards are seen by small suppliers and point-tool providers as enablers and facilitators, they are viewed by major EDA suppliers as "bad for business," because they limit the ability of the major EDA suppliers to control accounts. As a result, closed environments are still the norm. But the one definitive trend is away from the "frameworks" approach and towards "interoperability." This again is not surprising when one considers the fact that Intel spent significant effort and time to adopt a commercial framework but found itself still locked-in.

Traditional EDA architectures and process flows implied in most existing toolkits do not readily support "new" design processes based on integrated product development, concurrent optimization, and collaborative design. Finally, cultural and organizational issues inhibit rapid change and flexibility in use of EDA solutions.

Collett Survey

The recent Collett Survey has several interesting results that have both technical and business implications.

- The ratio of engineers to CAD support staff is a key indicator of time to problem resolution. This ratio, which averages 8:1 in industry is 5:1 at semiconductor houses. The survey found that semiconductor houses have lower ratios, do more complex designs, and experience significant interoperability and performance issues but overcome them more quickly. Others doing less complex designs experience fewer interoperability issues, but take longer to solve them because they have higher ratios.
- On the average, <u>specification changes</u> stretch design cycle times by over 2 to 8 months while interoperability issues add only 1 to 2 months.
- Typical gate count that stands at $80K today is expected to grow to $160K in 18 months.
- Interoperability-related non-productive time associated with non-value-added activities such as moving files, running translators, accounts for 15% of the design effort -- a $4 billion problem!

- 75% of the users surveyed indicated that "out-of-the-box" EDA integration is either below average or unacceptable with time-to-market pressures and system performance goals creating the greatest amount of dissatisfaction.
- The most time consuming loops in the design cycle is logic design ... and the most often executed loop is logic design.
- The mean time to bring up a new tool is 17.5 days.
- While VHDL usage is growing, Venlog usage is not.
- For each dollar spent on tools, customers spend an additional $2.50 (on the average) for maintenance and support.
- Typical corporate EDA budget: 20% EDA tools; 40% EDA personnel-related expenses (e.g., salaries $773 million); 40% maintenance of EDA hardware and software.
- EDA personnel-related expenses breakdown: 22% administration; **29% management of methodology, technology, and processes**; 20% library management; 12% glue software development; 13% application software development.

EDA Business Model is in Transition
The EDA Business Model is evolving along each of the three key dimensions: The System Model; The Licensing Model; The Distribution and Support Model. We present these trends in Table A-1.

Table A-1.
Business Model Trends

Model Type	Traditional	New
System Model	Monolithic, fully-integrated, single supplier	"Plug and play"
Licensing Model	Node-locked or floating	Per-project or short-term lease; Pay for use
Distribution and Supplier Model	Direct sales; bundled maintenance and upgrades	VAR; Time and materials

It is also becoming increasingly clear that a revenue stream is needed to support new-product R&D. The bottom line is that the evolution of the new EDA Supplier Business Model can be influenced but only through our purchasing decisions.

Industry Trends
Platform strategy typically varies with the EDA vendor's tool positioning. Very high end and middle of the road tool vendors are expected to pursue Windows NT and traditional UNIX workstation. For example, Intergraph design tools are being offered on Windows NT whereas Mentor and Cadence continue to offer their software on UNIX workstations. Low end tools such as from Tanner Research will be offered on Windows platforms. In general, new small high-value solution providers that emphasize agility are expected to find common ground on NT first, then migrate to UNIX. Despite these developments, UNIX is expected to remain a platform of choice, but is expected to see increasing competition form lower cost, more interoperable solutions on Windows and Windows NT platforms.

Despite the fact that data management is necessary now more than ever before, few EDA vendors and uses have taken it seriously until now. Today, the EDA user industry acknowledges that development teams on large, complex projects have little chance to being successful without data

management. A successful data management strategy is absolutely essential t corporate success in a highly competitive, time-to-market driven enterprise. Despite the successes of product data management systems such as Sherpa, significant work remains ahead. Specifically, the enterprise scalability of product data management systems and their interface to design process management systems continue to be fuzzy.

EDA Outlook

Market "seat saturation" spells the end of an era characterized by high-dollar EDA suppliers. Growth is expected to occur in high-value provides characterized by low-cost, "best of breed," multi-platform solutions. Platform-supported interoperability is expected to overcome ultimately EDA-erected barriers. Real interoperability is expected to spur a surge in innovation in the 3 to 5 year time frame. Finally, EDA users are expected to close the EDA-technology gap with continued use and generation of new requirements.